

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

EMULATION OF THE AN/UYK-20
TACTICAL DATA COMPUTER
ON THE BURROUGHS D-MACHINE

by

Ralph Harry Anzelmo
and
Theodore Lawrence Kaye

March 1977

Thesis Advisor:

L. V. Rich

Approved for public release; distribution unlimited.

7 177953

REPORT DOCUMENTATION PAGE		LIBRARY NAVAL POSTGRADUATE SCHOOL	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) Emulation of the AN/UYK-20 Tactical Data Computer on the Burroughs D-machine		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1977	
7. AUTHOR(s) Ralph Harry Anzelmo and Theodore Lawrence Kaye		8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 1977	
		13. NUMBER OF PAGES 263	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified	
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) emulation microprogramming AN/UYK-20 Burroughs D-machine			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A representation of the Univac AN/UYK-20 computer systems has been emulated on the Burroughs D Interpreter-Based System. The entire AN/UYK-20 instruction repertoire has been emulated with the exception of the "math pac" option (floating point arithmetic and cordic functions), clock and interrupt codes, and input/output operation codes. Modular design with extensive documentation has been implemented throughout program (cont.)			

20. (cont.)

development allowing for ease of modification and further extensions to the existing emulation. Emulation and hardware architecture of the AN/UYK-20 and the Burroughs D-machine, are discussed in conjunction with the AN/UYK-20 itself. Methods of testing and debugging, sample test programs and recommendations for continued design modifications to the emulation are presented.

Emulation
of the
AN/UYK-20
Tactical Data Computer
on the
Burroughs D-machine

by

Ralph Harry Anzelmo
Captain, United States Marine Corps
B.A., Montclair State College, 1968

Theodore Lawrence Kaye
Lieutenant, United States Navy
B.S.S.E., United States Naval Academy, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
MARCH 1977

ABSTRACT

A representation of the Univac AN/UYK-20 computer system has been emulated on the Burroughs D Interpreter-Based System. The entire AN/UYK-20 instruction repertoire has been emulated with the exception of the 'math pac' option (floating point arithmetic and cordic functions), clock and interrupt codes, and input/output operation codes. Modular design with extensive documentation has been implemented throughout program development allowing for ease of modification and further extensions to the existing emulation. Emulation and the hardware architecture of the AN/UYK-20 and the Burroughs D-machine, are discussed in conjunction with the AN/UYK-20 emulation itself. Methods of testing and debugging, sample test programs and recommendations for continued design modifications to the emulation are presented.

CONTENTS

I.	INTRODUCTION.....	9
A.	STATEMENT OF THE PROBLEM.....	9
B.	APPLICATIONS OF THE AN/UYK-20	10
C.	PROJECT DESIGN OBJECTIVES.....	11
II.	EMULATION.....	13
A.	HISTORICAL BACKGROUND.....	13
B.	MICROPROGRAMMING.....	16
C.	THE GOALS OF EMULATION.....	21
D.	EMULATION VERSUS SIMULATION.....	21
E.	EMULATION TECHNIQUES.....	23
F.	EMULATION HARDWARE.....	25
III.	AN/UYK-20 ARCHITECTURE.....	28
A.	HARDWARE DESIGN.....	29
B.	INSTRUCTION FORMATS AND REPERTOIRE.....	38
1.	Repertoire of Instructions.....	38
2.	Instruction Format.....	41
IV.	BURROUGHS D-MACHINE.....	46
A.	HARDWARE DESCRIPTION.....	46
1.	Logic Unit.....	48
2.	The Control Unit.....	52
3.	Memory Control Unit.....	53
4.	Microprogram memory (M-memory).....	55
B.	NPS MICROPROGRAMMING FACILITY.....	56
1.	Physical Description.....	56

2. Input/Output Interface.....	58
3. Memory Interface.....	59
C. MICROINSTRUCTION TIMING.....	60
D. TRANSLANG.....	64
V. AN/UYK-20 EMULATOR.....	66
A. EMULATION DESIGN.....	66
1. Functional Components.....	66
2. Main Memory Organization.....	68
3. Emulation Program Status Word.....	70
4. D-Machine Registers.....	73
B. LOADER.....	75
C. THE FETCH MODULE.....	77
D. OPCODE IMPLEMENTATION.....	83
E. UTILITIES.....	86
F. INPUT/OUTPUT CONTROLLER.....	87
VI. EMULATION TESTING.....	91
A. METHOD OF TESTING.....	91
B. SAMPLE TEST PROGRAMS.....	94
C. TEST RESULTS.....	95
VII. SUMMARY AND RECOMMENDATIONS.....	97
A. EXPERIENCE WITH HARDWARE.....	97
B. LESSONS LEARNED.....	98
C. EMULATION PROBLEMS.....	99
D. RESULTS.....	100
E. RECOMMENDATIONS AND FOLLOW-ON TOPICS	101
APPENDIX A. AN/UYK-20 EMULATOR USER'S MANUAL.....	103

APPENDIX B. LOADER CONTROL CARD FORMATS.....	107
APPENDIX C. AN/UYK-20 EMULATOR INSTRUCTION FORMAT.....	110
APPENDIX D. SAMPLE DEBUGGER OUTPUT.....	111
APPENDIX E. SAMPLE TEST PROGRAMS.....	113
APPENDIX F. EMULATOR LISTING.....	120
BIBLIOGRAPHY.....	259
INITIAL DISTRIBUTION LIST.....	262

ACKNOWLEDGEMENTS

Our sincere gratitude is expressed to Lieutenant Lyle V. Rich, SC, USN for sponsoring this thesis and without whose guidance this research would not have been a success. Technical expertise for the AN/UYK-20 was provided by John H. Westergren, Field Engineer, Mini-Systems Support Operations, Sperry Univac Computer Systems, St. Paul, Minnesota. Initial instruction on the operation and design of the Burroughs D-machine was graciously provided by Lieutenant Jerry M. Haggerty, USN and Lieutenant John M. Hartling, USN. Finally, the authors would like to dedicate this thesis to their wives, whose love, devotion, and understanding enabled this project to be completed.

I. INTRODUCTION

A. STATEMENT OF THE PROBLEM

The Navy has been challenged with maintaining the newest, most efficient tactical data systems consistent with the continually increasing demands and requirements of the real-time environment. There is an extensive conversion effort required to change from existing systems to newer more sophisticated technology such as the AN/UYK-20. Inherent in upgrading to a new system is the complex software redesign and modification process which is often hindered by the absence of the new computer system.

Unfortunately, the demands of a military installation require software generation prior to implementation of an upgraded computer system. One solution to this problem is to utilize for software development an intermediate computer system which has the capability of emulating the anticipated target machine. This provides a vehicle for software design, development, and testing prior to transitioning to the new system.

Currently, the Naval Postgraduate School (NPS) Computer Science Department maintains a Burroughs Interpreter-Based System known as the Burroughs D-machine. The D-machine is capable of being microprogrammed to emulate any of a myriad

of target machines. It effectively enables students to create their own computer knowing only the machine instruction repertoire for the control unit in the target machine.

The problem presented was to develop a feasible working model of the AN/UYK-20 on the microprogrammable Burroughs D-machine. The project provided an opportunity to obtain practical experience with contemporary hardware and to manipulate writeable control stores to imitate a Navy tactical computer.

B. APPLICATIONS OF THE AN/UYK-20

The Univac AN/UYK-20 minicomputer is a general purpose militarized digital computer adaptable to numerous tactical applications. The AN/UYK-20 has been successfully utilized in many time-critical, real-time systems including fire control radar, communication controllers, signal processing analyzers for sonar and beacon signals, and numerous weapons control systems. A subsequent chapter will be devoted to the technical aspects and internal design of the AN/UYK-20.

Because of its size, ruggedness, and computing capabilities, the AN/UYK-20 has been designated the Navy's standard tactical minicomputer [1b]. It was selected for emulation in order to provide a feasible platform for software development to those military installations either contemplating or in the process of receiving an AN/UYK-20.

A workable emulation would allow military applications such as data reduction, navigation, telemetry, sensor processing, range tracking and logistics to have software packages developed, tested, and modified prior to arrival of the AN/UYK-20. Furthermore, it would permit personnel to become familiar with the machine by providing advanced training, thereby easing the transition phase to the new system.

C. PROJECT DESIGN OBJECTIVES

Several design techniques were used throughout the development of this project: 1) modularity, 2) structured programming, and 3) extensive documentation. These design features will aid the interested reader as well as simplify any future extensions or modifications to the existing emulation.

Modular design was utilized by creating independent program segments which were individually developed, debugged, and tested. These modules or subroutines provided a strong foundation which were readily modified throughout the entire programming effort. Conceptually, the emulation was divided into relatively small entities which were further reduced to program segments rarely exceeding one page in length.

Structured programming was demonstrated by utilizing a limited number of control flow structures and maintaining a common logical design throughout the entire emulation. Comprehensible code held precedence over extremely efficient

code.

In addition to modularity and structured programming, the entire programming endeavor was supplemented with extensive commenting to provide the necessary self-documentation to promote and facilitate program translation, modification, and fusing with the other independent modules. These concepts promoted the extensive team effort required to achieve the research goals. In addition, it will provide ease of program maintenance and modification in the future.

II. EMULATION

A. HISTORICAL BACKGROUND

The term microprogramming was first utilized in an article by Professor M. V. Wilkes of the Cambridge University Mathematical Laboratory in 1951 [24]. His paper concentrated on a control section within the computer which, when programmatically controlled, performed register-to-register data transfers sequentially and in parallel for the execution of a single machine instruction. A sequence of operations (microinstructions) required for execution of a machine instruction is considered a microprogram.

Traditionally, the computer has been composed of essentially five components: the arithmetic/logic unit, the control unit, memory or storage, input, and output (Figure II-1). The control section sets the proper conditions for the opening and closing of required gates in the logic network. Historically, the control section has been hardware consisting of a series of decoders and flip-flops along with their associated circuitry. Therefore, every machine instruction had a fixed interpretation which was hardwired within the control unit.

In 1957, Wilke's definition of microprogramming was slightly modified. It was defined as a technique of design-

ing the control circuits of an electronic digital computer to interpret and execute a given set of machine operations as an equivalent set of micro-operations [15].

The hardwired control section can be modified by interchanging ROM modules or other hardware components, by replacing the control section with a programmable (dynamically writeable) control store which in itself is a separate word-organized memory (Figure II-2) or by combining both approaches. A programmable control store allows rapid changes in the machine's instruction repertoire while maintaining maximum design flexibility. The resulting computer system is microprogrammable and capable of storing a series of changeable machine personalities.

The computer control store can thus be modified to allow the execution of machine language programs intended for a variety of machine architectures. This process can be compared to replacing hardware components found in conventionally designed computer systems. The primary advantage of microprogrammed logic is the capability to perform various control sequences without hardware modifications. The process through which the hardware components of one machine (host) are made to imitate the specific hardware characteristics of another machine (target) is known as emulation.

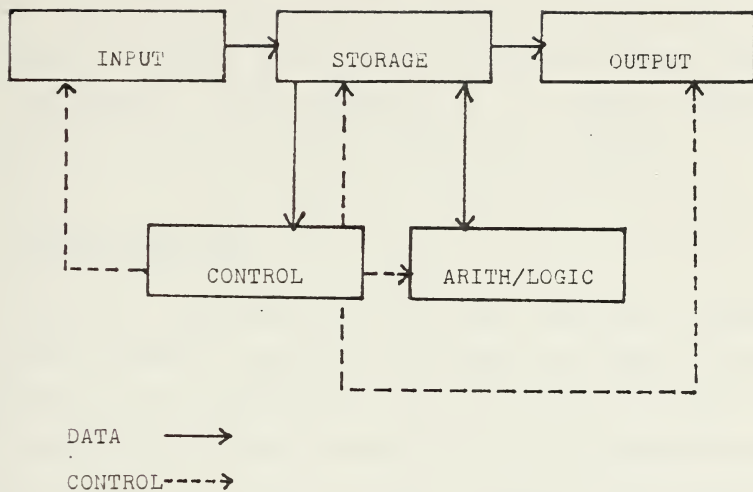


Figure II-1 (11)

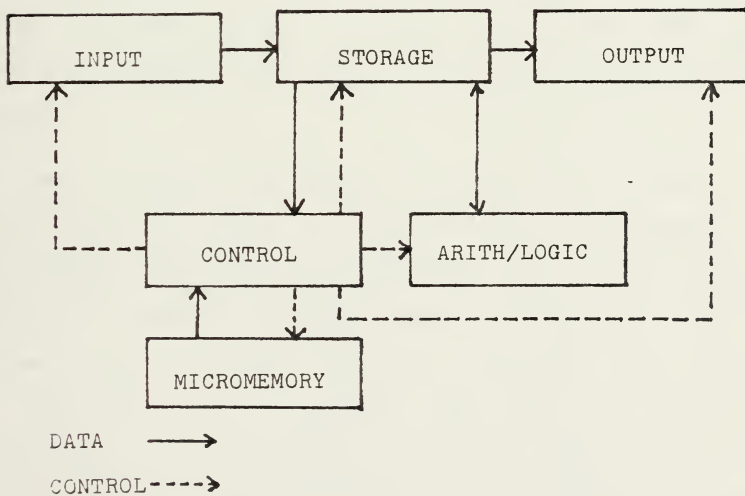


Figure II-2

Emulation allows the computer scientist to create various machine architectures from a single microprogrammable host. The complete set of microprograms (firmware) and the necessary hardware, as well as the required software, added to one computer system enabling it to execute programs designed for another system is known as an emulator.

B. MICROPROGRAMMING

Computer manufacturers have made available numerous microprogrammable machines which permit the user to tailor his instruction repertoire to meet the needs of his particular application. Some examples of microprogrammable computer systems are the Burroughs D-machine, the Nanodata QM-1, the Varian 73, the Standard Logic CASH-8, or the Hewlett-Packard 2100. These microprogrammable systems provide the benefits of flexibility, lower system costs and a systematic approach to system design if utilized effectively.

When a manufacturer designs a dynamically writeable control store, the amount of parallelism to be allowed must be determined. Parallelism is defined to be the simultaneous control of numerous hardware resources. There are basically three forms of control: vertical, horizontal, and residual. In vertical microprogramming, each instruction controls a single operation with program flow being sequential, unless the instruction was a conditional or unconditional branch. By contrast, a horizontally microprogrammed machine is

programmed via instructions which simultaneously control multiple resources including condition testing and microinstruction sequencing.

Horizontal microinstructions usually are not encoded which means each bit controls one machine resource or operation. They usually have a wider word than vertical instructions and consequently consume more memory. Vertical instructions are usually encoded with one or two levels. Encoding means the value of a control field in the microinstruction is a binary code specifying which resource or operation is to be performed. The horizontal microinstructions have the potential of being much more efficient resource managers and consequently are more difficult to optimally design than their vertical counterparts.

Combining the attributes of horizontal and vertical microprogramming results in residual control. This method saves memory by using vertical microinstructions while simultaneously controlling multiple parallel resources via setup registers.

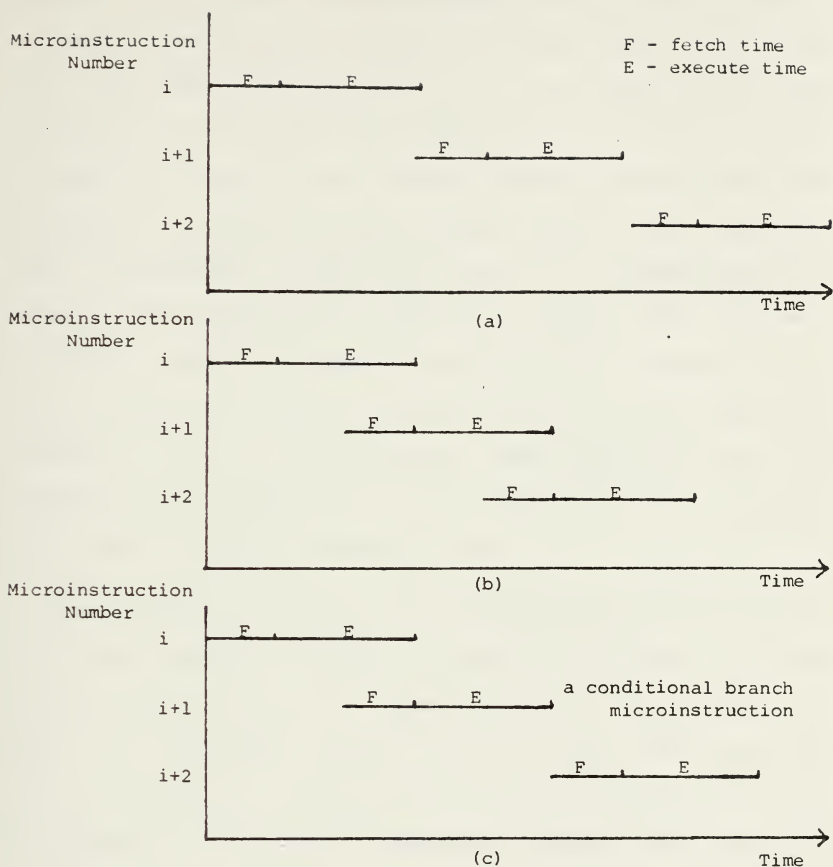
Microinstruction implementation severely effects the speed of microprogram execution. In serial implementation, one microinstruction is fetched and fully executed prior to fetching the next instruction. This technique offers the advantage of logical simplicity while suffering from lack of efficiency since it consumes the maximum amount of time.

'Parallel' implementation permits fetching of the next

instruction before termination of the previous instruction. The obvious advantage is execution speed which is of utmost importance when emulating another machine (Figure II-3) [2].

Another significant microprogramming characteristic is the number of phases used in the execution of each microinstruction. A monophase system means there are no subdivisions of the basic clock pulse and consequently each microinstruction is controlled by the transmission of the leading edge of a clock cycle. In a polyphase implementation scheme, the basic clock cycle is subdivided into minor phases which are independently generated via hardware. Although polyphase operations are more complex and require complicated control, they do permit faster resource manipulation when they are efficiently coded by allowing multiple operations to be performed during the same phase(s).

The microprogrammability of a given computer and the capabilities of its associated microprogramming language are directly effected by the the presence or absence of each of the alternative microprogramming characteristics described above. The microprogramming language spectrum ranges from the lowest level or microlanguage through the assembly languages to the high level procedural languages.



- (a) Serial fetch and execute.
- (b) Parallel fetch and execute.
- (c) Combined serial-parallel where next address depends on conditions in present cycle.

Figure II-3 [2]

The problems of microprogramming can be significantly reduced if suitable software support exists and is readily available. This support is usually in the form of simulators and debuggers. Typically, a simulator provides an alternative to assembly level coding by permitting the user to code in a higher level language and yet achieve the same results at the expense of some added memory and execution time. Debuggers are extremely useful in the developmental stages of microprogramming especially for new and experimental system design. Debuggers permit dynamic access to the machine status and register contents at the instant they are employed, i.e. a trace feature. Some debuggers offer the opportunity of assembling in-line. This option can drastically reduce required debugging time.

The primary application of microprogramming is to implement the necessary control structure required for the analysis and execution of machine level instructions by means of programmed control stores rather than hardwired logic. Therefore, a dynamically microprogrammable computer can provide a software development system which can be a cost-effective approach to experimentation with potential candidates for replacement computer systems or the design of completely new systems to fit the needs of unique applications.

C. THE GOALS OF EMULATION

A well-designed emulation can provide an opportunity to experiment and create software for new computer systems before the actual hardware is available. The utilization of an emulator can almost eliminate reprogramming, consequently smoothing the system transition period. In addition, emulation has provided a workable model of new systems under consideration for procurement, providing a much more detailed cost-benefit analysis of system conversion.

Furthermore, it is often economically sound to emulate a second generation computer with a third generation system. This provides growth to a contemporary system while fulfilling the requirements of the past in a cost-effective manner. However, this can be a disadvantage if the programming staff uses the emulation as a link to the old system and consequently fails to take advantage of the attributes of the new system.

D. EMULATION VERSUS SIMULATION

To accomplish the emulation objectives, certain design features must be incorporated into an emulation. Naturally, execution time and allocated memory are the two foremost considerations. Traditionally, the concept of mimicking another computer has been accomplished by either a simulator or an emulator, two concepts often confused with one another.

A simulator is a series of high level language (HLL) or assembly language statements which individually do not behave like the target machine instructions. The host machine executes its own native instructions in order to imitate the target machine operations. Consequently, simulation is a rather slow technique because it requires an intermediate translation. In addition, simulation of certain instructions such as bit manipulation and shifting operations can require an enormous amount of intermediate code generation demanding a significantly larger memory allocation.

An emulator is a microprogram that is executed on the host machine, performing machine instructions of the target machine. Since an emulator accepts the binary object code of the target machine and directly executes these instructions, it can be extremely efficient in terms of time and space requirements. The execution time of an emulation is dependent upon many factors: clock rates of the two machines, frequency of memory references, high speed shifting compatibility, required register mapping between target and host machines, bit manipulation capability of the two machines, condition code selection and testing, flexible data path selection capability, interrupt similarities, input/output compatibility, and microprogramming efficiency. If the hardware features between target and host machines are extremely compatible and highly efficient microprogramming has been employed, an emulation performance ratio (host

to target) of nearly one to one can be attained. This emulation performance ratio (EPR) has been demonstrated by the emulation of the SKC-2070 on the Nanodata QM-1 computer. It is possible to achieve an EPR better than one to one under ideal situations, when the host machine has a much faster internal operation execution rate [1].

Several distinct advantages can be realized using emulation as compared to simulation. The execution speed is significantly better by at least an order of magnitude. The target machine representation in firmware is closer to the actual hardware design and total access to the lowest machine level is achievable. Perhaps most noteworthy, emulation provides the opportunity to rapidly create test beds for numerous machine architectures and provide a basis for new system development.

E. EMULATION TECHNIQUES

Traditionally, there have been three approaches for emulating machine instructions: 1) hardware or firmware assistance to a software simulation as demonstrated by the IBM 360/b5 emulation of the IBM 7090, 2) independent host system hardware or firmware which provides for complete execution of the target machine's instruction repertoire of which the Burroughs D-machine emulation of the AN/UYK-20 is an example, and 3) an auxiliary processor which is operated in conjunction with the host machine to execute target machine instructions [14].

Software-controlled emulation is usually characterized by categorizing the target machine instructions into three distinct classes: easily emulated instructions, complex instructions not readily emulated, and those instructions not deemed necessary for the desired application. Instruction usage is significant in this classification process. Each class of instructions becomes a candidate for direct hardware or firmware implementation. The first emulated function in this approach is usually the fetch and analysis operation. After the instruction is analyzed, the appropriate opcode subfunction can be executed.

An alternative emulation technique is the firmware-controlled method. This approach is identified by having system control reside completely in firmware or hardware during the emulation process. All instructions are executed on the host machine as if they were indigenous to the target machine. This method is much more efficient than the software-controlled technique; however, it is more expensive and the cost differential is directly related to the required performance level. Performance is dependent upon the number of required data paths, arithmetic units, and other additional logic circuitry which must supplement the host machine architecture.

Upon entering the emulation mode in a firmware-controlled emulator, the machine performs like the target machine until encountering an exit situation. There exists three exit modes: 1) priority interrupt, 2) not implemented

instruction, and 3) deliberate exit because of a debugging routine.

The third emulation technique consists of utilizing auxiliary hardware electronically attached to the host computer for the sole purpose of executing target machine instructions. In effect, a target machine is composed of host machine hardware with the necessary additional components required to create an effective emulator.

F. EMULATION HARDWARE

The development of writeable control stores and microprogramming techniques have significantly influenced computer design. This section will describe some of the available dynamically microprogrammable hardware (Figure II-4).

The Hewlett-Packard 2100 is a general purpose minicomputer. It has a unique control store divided into two segments. One section is ROM and the other section is user programmable. The machine is vertically microprogrammed using a standard 80 instruction machine language repertoire. A debugger and assembler assist the user in microprogram development [2].

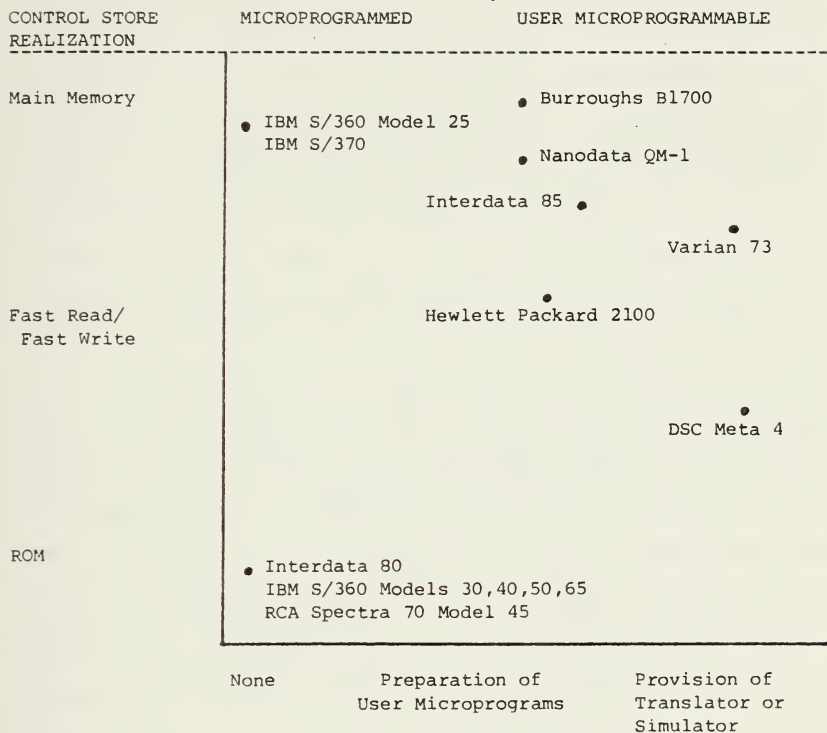
The Standard Logic CASH-8 is a high speed digital controller with a separate control store. It consists of 16 general purpose registers and an accumulator. The CASH-8 is vertically microprogrammed but does not support any language

above microlanguage [2].

The Varian 73 is a general purpose minicomputer that has a 150 instruction set. The horizontal microinstruction consists of 64 bits with 25 fields, some of which indicate register transfers, ALU operations, shifting, control store addressing, condition testing, I/O control and memory operations. The Varian 73 contains both a ROM control store and a writeable control store loadable from main memory. A microprogram assembler and interactive simulator are available [2].

The Nanodata QM-1 is unique in that it contains both a control store and a nanostore which are both loaded under user program control. The 18-bit vertical microinstructions are stored in the control store, fetched and then interpreted under nanoprogram control. A horizontal nanoinstruction is 360 bits which is subdivided into five 72-bit vectors. Assemblers for both microprograms and nanoprograms are available [2].

The previously described machines represent a small sample of the available microprogrammable computer architectures. The availability and flexibility of these computer systems has stimulated demand for these devices. Consequently, hardware manufacturers have been compelled to produce writeable control store equipment to satiate the needs of the computer market.



SUPPORT AVAILABLE TO USERS

Note: Relative microprogrammability is the distance from the origin to the machine point in two space.

Figure II-4 [2]

III. AN/UYK-20 ARCHITECTURE

Constructing an efficient emulation requires a precise understanding of the architecture and performance characteristics of the machine being emulated. An emulation must attempt to match the target machine's features and maintain its flexibility of hardware design as closely as possible. Although it is not required, an operational demonstration of the emulated machine can solve many emulation questions.

In emulating the AN/UYK-20, architecture and performance criteria were derived from technical publications, since an actual machine was unavailable. When inconsistencies appeared in the documentation, specific questions were posed to a UNIVAC field engineer, who often tested programs on the AN/UYK-20 to resolve inconsistencies. Documentation coupled with an expert consultant provided sufficient information for emulating the AN/UYK-20 successfully.

The intent of this chapter is to outline those features of the AN/UYK-20 significant to the emulation. A detailed hardware description can be found in Refs. 20, 22, 28.

A. HARDWARE DESIGN

The AN/UYK-20 was designed for the Navy to fulfill the requirements for small or medium size general purpose data processing in shipboard, mobile shelter, or other military environments. Sperry Univac incorporated minicomputer technology in constructing the AN/UYK-20, including MSI circuitry design, microprogrammed control, memory modularity, and asynchronous or synchronous input/output channels.

The AN/UYK-20 had to be extremely flexible in its applications, offering a wide range of configuration possibilities which were derivatives of the basic design. Modularity, a concept highly desirable in a military environment, was achieved by offering options that could be easily added using printed circuit cards and/or memory modules.

The AN/UYK-20 can accommodate up to eight 8K, sixteen-bit word boards of magnetic core storage with an access time of 750 nanoseconds. The central processor is controlled by a programmable micromemory which can be expanded by an additional 512 words. The microprogram controller is programmed at the factory, but the additional micromemory option is user defined. Both sections of micromemory are programmed using fusible links, and once programmed they are completely static (Figure III-1).

A memory interface is responsible for the transfer of data to memory from the central processor (CP) and input/output controller (IOC). Both the IOC and the CP are

capable of accessing all of memory (65,536 words maximum). The addition of direct memory access (DMA) provides a second memory interface and an additional access port which is connected to each of the two 32K memory segments.

The input/output controller permits the central processor to communicate with the external devices without interfering with program execution. The IOC has a maximum of 16 parallel or serial channels. Parallel data transfer takes place asynchronously using 8-bit, 16-bit, or 32-bit transfers. Serial interfaces are either synchronous or asynchronous, with word-to-serial or serial-to-word conversions occurring in the IOC. The IOC and CP compete for memory access through the memory interface with priority given to the IOC in the event of a simultaneous request. The IOC is permanently assigned several memory addresses for command word and interrupt word storage.

The addressable high-speed registers available in the AN/UYK-20 include the program address register (P-register), two 16-bit status registers (SR1 and SR2), a real-time clock register (32-bits), a monitor clock register (16-bits), and a set of sixteen 16-bit general registers. An additional stack of 16 general registers is an available hardware option.

The sixteen general registers were included to enhance the speed and performance of the AN/UYK-20, allowing most programs to use a great proportion of register-to-register

instructions. These general registers can be used as accumulators for arithmetic, shift, or logical functions, as index registers, or as temporary storage locations. The second set of general registers can be readily employed via a status bit. This status bit designates which general register stack is to be utilized. The duplicate set of general registers yields dividends in a multi-task or heavy-interrupt processing environment. This additional register set can be used to provide high-speed temporary storage, thus avoiding slower main memory storage of working variables.

The two 16-bit status registers and the program address register represent the machine status of the AN/UYK-20. When these registers are collectively referenced, they are called the program status word (48-bit PSW). The P-register indicates the next instruction to be executed. This instruction may be a 16-bit single-word instruction or a 32-bit double-word instruction. Program control can be modified by using an instruction which manipulates the contents of the P-register.

Status register 1 contains bit information concerning condition code settings, overflow, and carry bits, interrupt codes, and numerous other machine indications (Figure III-2).

AN/UYK-20 FUNCTIONAL ARCHITECTURE

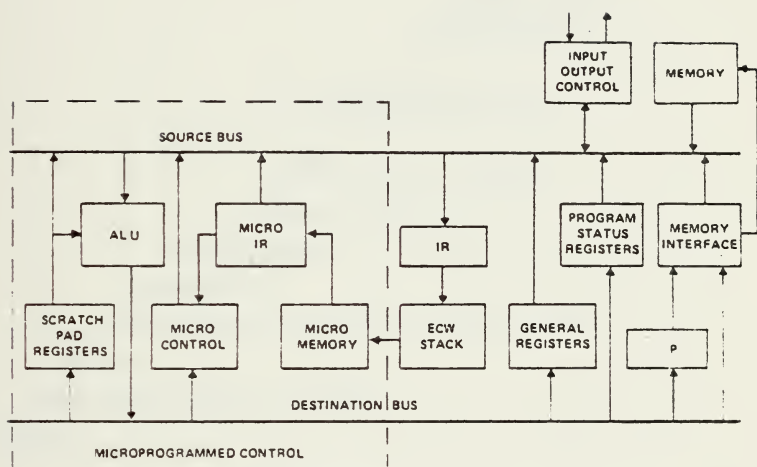
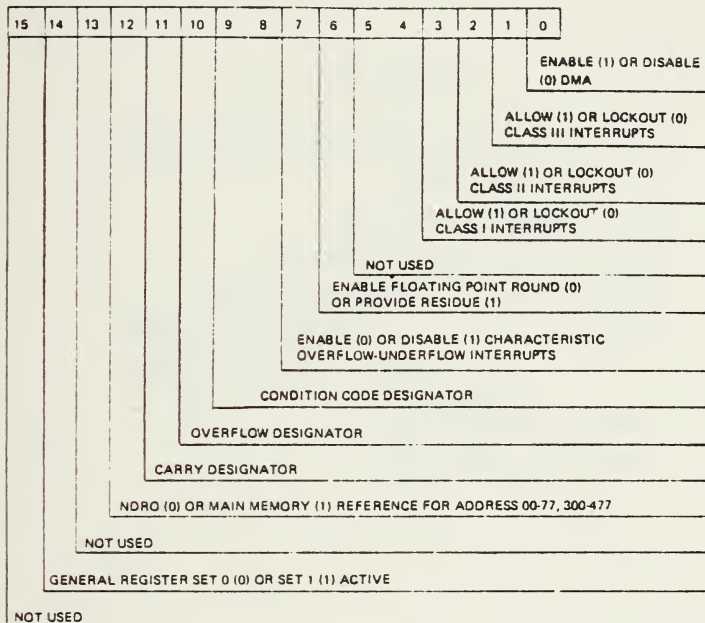


Figure III-1 [22]

STATUS REGISTER 1



CONDITION CODES

SR BITS 8 and 9	ARITHMETIC	COMPARE
00	0	$(R_a) = (R_m)$ or (Y)
01	>0 (POS)	$(R_a) > (R_m)$ or (Y)
10	Not Used	Not Used
11	<0 (NEG)	$(R_a) < (R_m)$ or (Y)

Figure III-2 [23]

Status register 2 holds control bits for direct or indirect addressing, and holds interrupt codes. Interrupt processing routines set bits in the interrupt code field corresponding to the IOC interrupt (Figure III-3).

STATUS REGISTER 2

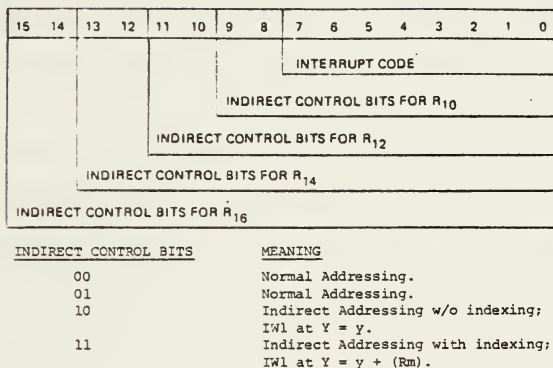


Figure III-3 (23)

The real-time clock and monitor clock registers provide program-controlled interrupt capability which is useful for timing and synchronizing program segments with real-time events. The real-time clock (RTC) is a 32-bit register used as count-up storage while the monitor clock (MON) is a 16-bit count-down register. A one kHz internal oscillator controls the counting speed of both registers. An optional

external clock operating at a frequency up to 50 kHz is also available.

Interrupt processing in the AN/UUK-20 is conducted using a priority level scheme which classifies interrupts into three priority levels (classes). Interrupts within the same class are assigned a priority ranking and a code which identifies which processing routine to execute. During interrupt handling, all interrupts of the same level or lower level are locked out until the CP is completed processing the current interrupt. Higher priority interrupts can override the lockout and cause the CP to honor them first, holding the lower level interrupts in abeyance until higher level interrupt processing is completed. The highest priority interrupts are hardware malfunctions, followed by software interrupts, and at the lowest level, IOC interrupts.

Permanent locations in memory corresponding to each interrupt class hold the PSW and RTC when an interrupt is being honored. Likewise, other permanent memory addresses assigned to each interrupt class hold the appropriate interrupt routine entrance location to be loaded into the PSW.

Memory addressing is accomplished using 64 page address registers which separate memory into 1024-word pages. Absolute addresses are formed by isolating the upper six bits of the relative address to find the page address register number, and then concatenating the lower six bits of the

page address register contents with the lower ten bits of the relative address (Figure III-4). Any operation that stores into memory sets the most significant bit of the page address register used in generating the address.

Some additional hardware features of the AN/UYK-20 are those functions available on the maintenance panel of the machine itself. These include a breakpoint feature which allows an operator to insert from the panel an address which causes the AN/UYK-20 to stop execution when the selected address is referenced. Other available toggles allow halting execution programmatically using Key 1 or Key 2 on the maintenance panel. These additional hardware features are useful debugging tools.

MEMORY ADDRESS GENERATION

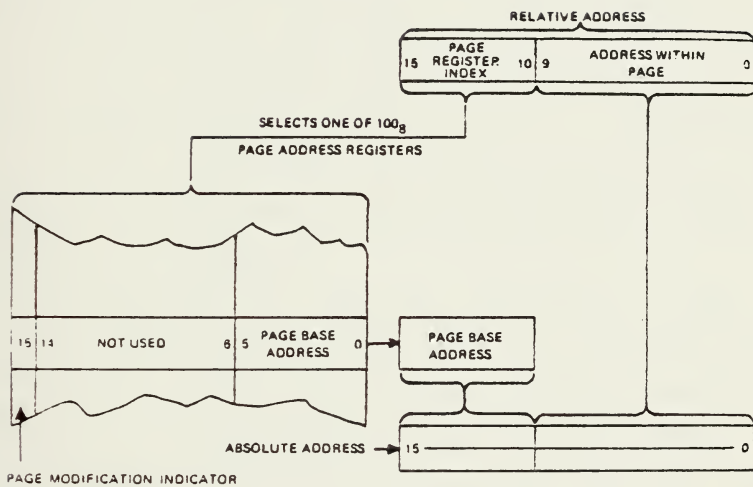


Figure III-4 [22]

B. INSTRUCTION FORMATS AND REPERTOIRE

1. Repertoire of Instructions

The AN/UYK-20 instruction set is composed of nearly 260 separate instructions designed to be both versatile and comprehensive. Both single-word (16-bit) and double-word (32-bit) instructions are available. Some of these instructions are specifically designed to meet the requirements of a real-time environment. A few sample instructions include:

- a. Local jump - used to facilitate loops, saving several steps in program execution.
- b. Reverse register - used in a communication environment when data is received in one sequence but must be transmitted in reverse sequence.
- c. Set bit, clear bit, and test bit - used to test individual bits in registers saving considerable execution time in programs that communicate via flags and status words.

Additional flexibility is provided when the 'math pac' hardware option is included in the AN/UYK-20 configuration. Some 33 additional opcodes are added to the instruction set in order to increase the computational capabilities of the machine. An instruction for square root, double precision multiply/divide instructions, as well as hardware trigonometric and hyperbolic functions which utilize coordinate conversion algorithms (cordic) are included.

Single-word instructions are generally employed when manipulating operands in high-speed registers. Double-word instructions are used in operations involving direct or indirect addressing with or without indexing, or supplying 16-bit constants to programs. Typical instruction speeds for a single-word versus a double-word instruction are:

	SINGLE-WORD	DOUBLE-WORD
ADD	.75 - 1.5 usec	1.5 - 2.25 usec
LOAD	.75 - 1.5 usec	1.5 - 2.25 usec
MULTIPLY	3.8 - 4.0 usec	4.4 - 4.6 usec
DIVIDE	6.8 - 7.0 usec	7.4 - 7.5 usec

Nearly all instructions affect condition bits in status register 1. The AN/UYK-20 sets these bits as a result of two types of operations. Most instructions that do not involve compare logic are categorized as arithmetic instructions. When the result of the arithmetic operation is determined and is about to be stored in memory or a register, a condition code is set indicating whether the result is positive, negative, or zero. Compare instructions set the condition bits based on a greater than, less than, or equal to comparison of two registers or a register and the contents of a memory address.

Two other bits in SR1 are set as a result of computational or shifting instructions. The overflow designator is set whenever an arithmetic or shift operation requires more bits than are provided for in a register (16 bits).

The carry designator is set when an arithmetic operator generates a carry beyond the most significant bit of the register.

The AN/UYK-20 allows five different types of operand formats: single-length, byte, literal, optional floating point, and double-length. Single-length operands are 16-bit values with the sign bit assumed to be in the most significant bit. In arithmetic calculations, the single-length word is assumed to be a two's complement integer. Byte operands are considered 8-bit unsigned integers, and can be the most significant or least significant half-word of a memory location. Literal operands are 4-bit unsigned integers denoted by the 'm' field of a literal type instruction. Floating point operands are formed using two adjacent registers or memory locations with fields containing the sign of the fraction, the characteristic, and 24 bits of the fraction.

Double-length operands are concatenated from two adjacent registers or two adjacent memory addresses. The most significant half of the double-length operand is contained in the low-order register or memory address, and the least significant half in the next sequential register or address. The sign bit for both words resides in the high-order bit position of the most significant half-word and when used in an arithmetic calculation, the double-length operand is treated as a two's complement integer. Instructions involving double-length operands require the low-order

register or memory address to be even, since the adjacent cell address is formed by 'OR'-ing a 1 in the least significant bit of the address.

2. Instruction Format

The AN/UYK-20 has five different instruction word formats, designated by two-letter mnemonic codes. These codes are: RR (Register-Register), RL (Register-Literal), RI (Register-Immediate), RK (Register-Constant), and RX (Register-Index). Each of these formats are designated in the instruction word by a combination of the opcode field and the subfunction code. Registers are identified in the 'a' and 'm' fields of the instruction, and are referred to by the notation Ra and Rm. The 'v' field is treated as an address or arithmetic constant, depending on the instruction (Figure III-5).

The format RR single-word instructions perform operations involving the registers specified by the 'a' and 'm' fields. No memory references are made to access operands, causing considerable savings in execution time. The 'a' or 'm' field may be used to index special operations on registers.

Format RL instructions use a 16-bit instruction word, using one or two general registers. The 'a' designator selects the general register Ra or register pair Ra, Ra + 1. The 'm' designator contains the 4-bit literal value which will be used in the instruction.

INSTRUCTION FORMATS

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RR	Op*				U				a				m			

a selects R_a ; m selects R_m .

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RL	Op*				U				a				m			

a selects R_a ; m contains 4-bit constants.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RI-1 Local Jump	Op*				1				d							

d is signed number of instructions to jump, relative to current position.
(+ is forward; - is backward)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RI-2 Indexed Memory	Op*				1				a				m			

a selects R_a ; m selects R_m . R_m selects memory address.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
RK	Op*							2		a							m																							
																		y																						

a selects R_a ; $m \neq 0$ selects R_m , $m = 0$ selects 0.
Operand = $y + R_m$ or 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX	Op*				3				a				m																			
																	y															

a selects R_a ; $m \neq 0$ selects R_m , $m = 0$ selects 0.
 $y + R_m$ or 0 selects memory address.

*Op is operation code

Figure III-5 [23]

RI format single-word instructions are separated into two categories. RI Type 1 instructions are local jump instructions where the P-register is increased or decreased by the 'd' field in the instruction. The 'd' field represents the two's complement deviation value and allows the P-register to be altered a maximum of +177 octal or -200 octal. RI Type 2 instructions involve operations that use the general registers Ra and Rm, and a memory address specified by the contents of Rm.

The format RK instructions contain 32 bits of information. The upper 16-bits contain the operation code and the designator fields. The 'a' field selects the general register Ra, and the 'm' field denotes how the next word, containing 'y', is to be used. If 'm' equals zero, then the effective operand address Y, equals 'y'. If 'm' does not equal zero, then Y is formed by adding the contents of Rm to 'y'.

Format RX are double-word instructions similar to the RK instruction that use direct and indirect addressing techniques determined by the 'm' field. If the 'm' field equals zero, then direct addressing without indexing is employed, with the effective operand address formed from the 'y' field itself. If the 'm' field equals 1 through 7, 11, 13, 15, or 17 (octal), then direct addressing with indexing is employed. The effective operand address is formed by adding the contents of Rm to 'y'. An 'm' field value of 10, 12, 14, or 16 (octal) indicates indirection is to be

employed, and the indirect address control fields in status register 2 contain information which is used to generate the effective operand address or a pair of indirect words. When the control fields equal 0 or 1, then direct addressing with indexing results. If the control field equals 2, then the contents of the first indirect word (IW1) is located at 'y'. Finally, if the control field equals 3, then IW1 is located at 'y' indexed by the contents of Rm. Indirect word format is shown in Figure III-6. Cascaded indirection is possible provided that the indirect words are properly encoded.

Byte addressing is accomplished using RX format instructions. A byte identifier (B) is used to determine which half-word (8-bit) is to be referenced. If B equals 0, then the most significant half-word in address Y is the operand byte. If B equals 1, then the least significant byte at Y is the operand. The least significant bit in the indexing register is used as the byte identifier, and the remaining 15 bits are used as the indexing value for finding Y. Indirect byte operand addressing formulae are included in Figure III-6. The following formulae apply for direct byte operand addressing:

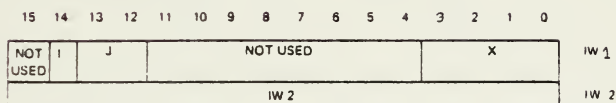
$$m=0, Y=y, \text{ and } B=0$$

$$m=1-7, 11, 13, 15, \text{ or } 17 \text{ octal}$$

$$Y=y + (Rm)/2 \text{ and } B=\text{LSB of } (Rm)$$

The preceding section has described the fundamentals of the AN/UYK-20 instruction formats. References 21 and 23 should be consulted for further information concerning instruction formats and decoding.

INDIRECT WORD FORMATS



OCTAL J VALUE	ADDRESS DETERMINATION
0	$Y = IW2$; if byte mode, upper byte is used.
1	$Y = IW2 + (Rx)$; if byte mode, LSB of R_x determines byte. *
2	$Y = IW2 + (Rm)$; if byte mode, LSB of R_m determines byte. *
3	$Y = IW2 + (Rm+1)$; if byte mode, LSB of $Rm+1$ determines byte. *

$I = 0$, DIRECT ADDRESSING, OPERAND AT ADDRESS Y

$I = 1$, CASCADED INDIRECT ADDRESSING, NEW INDIRECT WORD 1 AT ADDRESS Y

* To determine the operand address when in byte mode, the contents of R_x , or R_m , or $R_m + 1$ are right shifted 1 bit position and zero filled in the left most position

Figure III-6 [23]

IV. BURROUGHS D-MACHINE

A. HARDWARE DESCRIPTION

The microprogramming facility at the Naval Postgraduate School is composed of a Burroughs Interpreter-Based system. This system possesses the characteristic of being dynamically microprogrammable and is designed using a simple building block structure. A typical system is made up of a number of interpreters (processors), main memory modules, and input/output devices, along with a switch interlock device (SWI) controlling data flow on the data bus connecting the interpreters to main memory and peripheral devices. The heart of the system is the interpreter, also referred to as the D-machine.

A D-machine possesses five functional modules: the logic unit (LU), the control unit (CU), the memory unit (MU), nanomemory (NM), and microprogram memory (MPM). The system presently installed in the Computer Science Department combines nanomemory and microprogram memory into one functional unit. The architecture of the D-machine is designed around 8-bit word slices. Word lengths from 8 bits to 64 bits are permissible using the same functional unit (Figure IV-1).

INTERPRETER BLOCK DIAGRAM

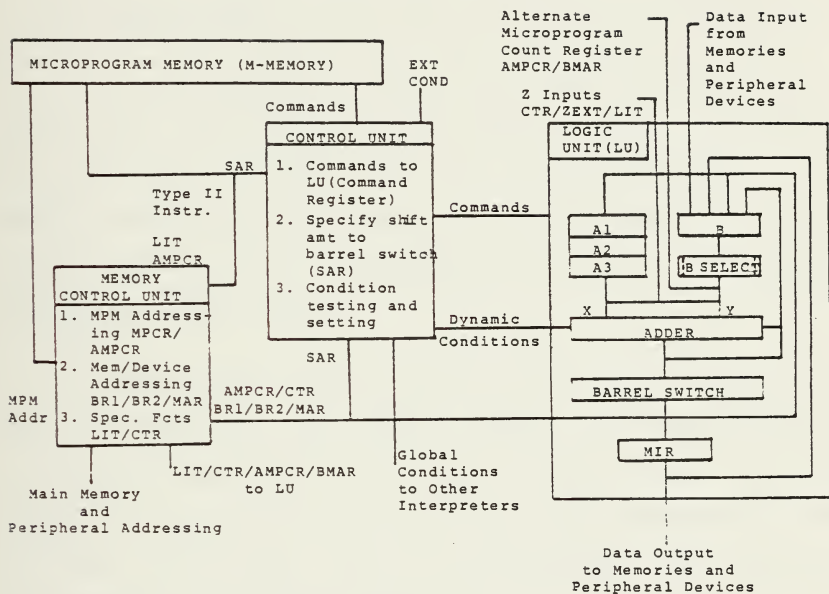


Figure IV-1 (a)

The D-machine used for this thesis has been configured as a 32-bit word processor. Reference 17 provides a thorough and concise description of the architecture of an interpreter-based microprogramming system. Reference 6 details the specifics of D-machine microprogramming which must be thoroughly understood by the programmer.

1. Logic Unit

The D-machine's logic unit performs shifting, arithmetic, and logic functions and contains scratch pad registers and data interfaces for the switch interlock. All adder operations are performed using two's complement arithmetic, and shifting is accomplished in a matrix of gates called the barrel switch.

The scratch pad registers A1, A2, and A3 are identical in function. They act as temporary storage registers within the D-machine and serve as primary inputs to the adder. The control unit determines which register will be an input to the adder. Any of the A-registers can receive the output from the barrel switch.

The B register functions as the primary external input interface from the switch interlock. It acts as the second input to the adder and can receive certain direct inputs from the adder's arithmetic operations. The b register can be loaded from any of the following sources:

- a. The barrel switch output.
- b. The adder output.
- c. External data from the switch interlock (or control panel switches).
- d. The memory information register (MIR).
- e. The complements of four bit or eight bit carries.
- f. The barrel switch ORed with the adder, external data from the switch interlock, or MIR.

The output of the B register is filtered prior to gating into the adder. The 'filter' consists of true/complement selection gates which are divided into three sections: the most significant bit, the least significant bit, and all the remaining central bits. The output of this filter is the independent result of these sections and may be either all zeroes, all ones, or the true contents or one's complement of the contents of the respective bits of the B register.

The memory information register is the interface to the switch interlock. MIR functions as a buffer to main memory or to a peripheral device. It is an output destination of the barrel switch and its output can be sent to the B-register or the switch interlock.

The adder in the logic unit performs all arithmetic functions and can be categorized as a version of a carry look-ahead adder. The control unit can gate various combi-

nations of A, B, and Z inputs to the adder. An 'A' input is defined as an input from one of the three scratch pad registers. A 'B' input is the output of the B register's true/complement filter gates. A 'Z' input is an external input to the logic unit and can originate from one of the following sources:

- a. The output of the counter in the memory control unit (MCU) which is gated to the most significant eight bits of the adder with the remaining bits zeroed.
- b. The output of the literal register in the MCU which is gated to the least significant eight bits of the adder with the remaining bits zeroed.
- c. An optional input which is gated into the middle bytes of the adder with the most and least significant bytes zeroed.
- d. The output of the alternate microprogram count register (AMPCTR) in the MCU which is gated into the least significant 12 bits of the adder (13 bits for 8K micromemory machines) with all other bits being zeroed.
- e. All zeroes.

Two inputs, selected from the A, B, or Z sources, are always gated to the adder. These inputs are referred to as X-select and Y-select. An X-select input may be either an A input or a Z input. If it is not specified, it is

assumed to be zero. A valid Y-select has either a B, Z or 1 as its input. Some Z inputs, however, are valid only as Y-select inputs. Any combination of valid X-selects and Y-selects are permissible addends, with an option of adding a one to the least significant bit. For subtraction operations, the value to be subtracted must be a Y-select; the Y-select input is subsequently two's complemented and gated to the adder. All binary Boolean operations between two adder inputs are allowed, and dynamic condition bits for overflow (AOV), all bits true (ABT), and most/least significant bit test (MST/LST) are available to the control unit for testing. Bit testing is a valuable feature for decoding instruction words.

The barrel switch is a matrix of gates that accepts parallel data from the adder and shifts the data any number of places to the left or right with zero fill. It also can right shift the word in an end-around fashion. The output of the barrel switch may be directed to any of the following destinations simultaneously:

- a. The A registers.
- b. The B register.
- c. The memory information register.
- d. The least significant 16 bits to the MCU registers.
- e. The least significant three to six bits to the control unit shift amount register (bit length depending on the word size of machine).

2. The Control Unit

The control unit has five major sections: the shift amount register (SAR), the condition register (COND), part of the control register (CR), the decoder for microprogram memory content, and clock control. This module of the D-machine manages the functions of the processor, and is directly responsible for logic unit operation.

SAR and its associated logic control shifting operations by loading shift amounts into SAR and generating the required controls indicated by the current microinstruction for the barrel switch. In addition, the SAR's logic generates the 'word length complement' of the SAR contents where the complement is defined to be that amount which would restore the bits of a word to their original position after an end-around shift of N followed by an end-around shift of the 'complement' of N. For example, if an end-around right shift of 20 was required in a 32-bit D-machine, another end-around shift of the complement of 20 (12) would be required to restore the contents to its original value.

The condition register has four major functions:

- a. It stores 12 resettable bits which are used as error indicators, interrupts, status, and lockout indicators.
- b. It selects one of 16 condition bits for performing conditional operations. These 16 bits are composed of the 12 condition bits of the

condition register plus the 4 dynamic conditions generated by the LU adder during the present clock time.

- c. It decodes bits from the memory for resetting, setting, or requesting the setting of designated bits of the condition register.
- d. It resolves priority between interpreters in the setting of global condition bits (GC), thereby providing a method of controlling inter-
interpreter lockout.

The control register stores the control bits of the 56-bit microinstruction that are not being used in the first phase of the execution cycle. The control register is subdivided into sections which are used by the memory control unit, the logic unit, and the control unit during the execution phase of a microinstruction. For a description of timing and phases, see section C of this chapter.

3. Memory Control Unit

The memory control unit provides the basic addressing interface between the D-machine and both main memory (S-memory) and microprogram memory (M-memory). One MCU can address 64K words (256K bytes) of main memory, and if the D-machine is configured with a second MCU, a maximum of 128K words can be addressed.

The memory control unit has three major sections:

- a. The microprogram address section controls the addressing of microprogram memory and the sequencing of microinstructions. It contains the microprogram count register (MPCR), the alternate microprogram count register (AMPCR), the incrementer, the microprogram address controls register, and their associated logic. For standard 4K M-memories, the MPCR and AMPCR are 12 bits long. For 8K M-memories, MPCR and AMPCR are 13 bits in length (the D-machines installed at the Postgraduate School employ 8K M-memories). AMPCR is a Y-select input to the logic unit adder.
- b. The memory/device address section contains the 8-bit memory address register (MAR), two 16-bit base registers (BR1 and BR2), output selection gates, and associated control logic. When forming a memory address, the lower eight bits of a base register and MAR are concatenated. The concatenated 24-bit contents of BR1/ BR2 and MAR (BMAR) is a valid Y-select input to the logic unit adder.
- c. The Z register section contains two registers which are Z inputs to the logic unit adder. The literal register (LIT) is an 8-bit register into which constants are loaded. An 8-bit counter (CTR) is used in conjunction with a counter overflow condition bit to control iterative

looping. The Z register section also contains selection gates for the loadable counter and its associated logic.

4. Microprogram memory (M-memory)

A D-machine may have a dual or single microprogram memory scheme. As indicated earlier, the D-machines used in this emulation project had a single microprogram memory, consolidating the microprogram memory and nanomemory into one 56-bit programmable store memory, often referred to as M-memory. Microprograms, consisting of 56-bit microinstructions, are dynamically changeable by the user, thus distinguishing the D-machine as an extremely flexible computing device.

The sequencing of microprogram instructions is controlled by a condition bit procedure which determines the successor command to be executed. M-memory provides data to the condition testing logic which then determines which condition is to be tested. The output of the condition testing logic is a true/false signal that is gated to the successor selection logic. This logic then selects between the three true and three false successor bits also provided by the M-memory word. The three selected bits provide eight possible successor combinations:

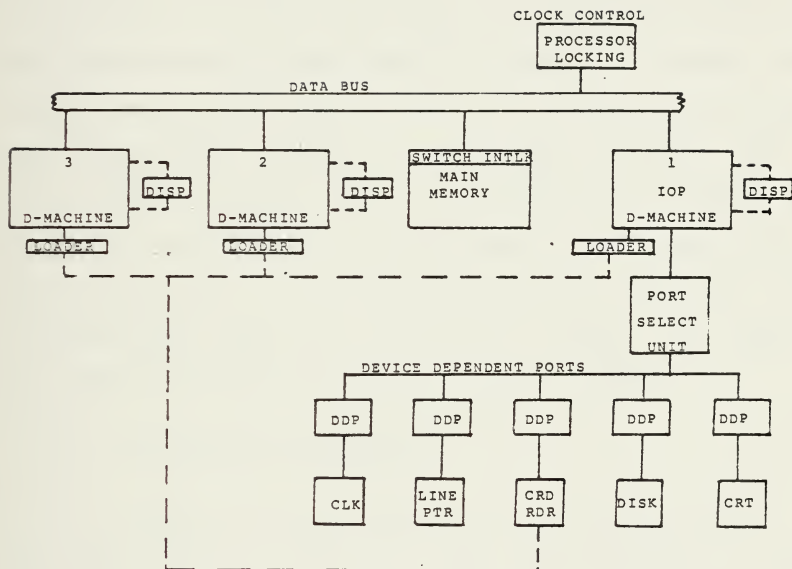
- a. WAIT Repeat the current instruction.
- b. STEP Step to the next instruction.
- c. SKIP Skip the next instruction.
- d. JUMP Jump to another M-memory address.
- e. RETN Return from a microprogram subroutine.
- f. CALL Call a microprogram subroutine.
- g. SAVE Step and save the instruction address.
- h. EXEC Execute one instruction out of sequence.

The particular successor command chosen controls gates which select the appropriate M-address from MPCR or AMPCR and provides incrementing logic for generating the next M-memory address. Except for the EXEC command, the MPCR is loaded with this M-memory address.

B. NPS MICROPROGRAMMING FACILITY

1. Physical Description

The Computer Science Department of the Naval Postgraduate School possesses a Burroughs Interpreter-Based System consisting of three interpreters (processors) also known as D-machines, a 64K 32-bit word, main memory module, a card reader, a dual cartridge disk drive, a line printer, and a Datamedia 2500 CRT functioning as a supervisor's console (Figure IV-2). All input and output is performed through a single D-machine processor, hardware configured with device dependent ports (DDP) for peripherals and the external clock.



NAVAL POSTGRADUATE SCHOOL MICROPROGRAMMING FACILITY
BURROUGHS INTERPRETER-BASED SYSTEM

Figure IV-2

After initial light-off and bootstrap, the system is configured into two Burroughs 6700 - LIFO ALGOL Stack-machines, each addressing 32K of memory and each communicating with the input output processor (IOP) in a pseudo-multiprocessing environment. Software, written in ALGOL, is provided which runs on the B-6700 system. A resident monitor control program and disk file manager control the maintenance of system files and the execution of jobs in a batch environment. Both D-machines compete for system jobs input from the card reader or CRT. Other software available includes an ALGOL compiler (a derivative of ALGOL 60), a microprogram translator called TRANSLANG, a line editor, and a simulation program for microprograms. TRANSLANG provides the medium by which microprograms are written. User microprograms loaded into a D-machine change its identity and destroy the Stack-machine previously loaded.

2. Input/Output Interface

Pivotal to the operation of the Burroughs system is the input/output interface. Only one processor, the IOP, communicates with peripherals, and the other D-machines, configured as Stack-machines, must compete for its services. The IOP communicates asynchronously, using a conventional 'handshake' method. Since all interpreters have access to the main memory module, a communications link has been established using the upper two 32-bit words of main memory. If a Stack-machine wishes to communicate with the IOP, it places a message into address 65,535 known as the 'mailbox',

and issues an interrupt (INT) to the IOP. The IOP periodically tests INT, and if set, will retrieve the contents of the mailbox (and mailbox + 1 if required) and perform the desired operation. The other Stack-machine is locked out from interrogating the IOP until it has completed processing the request. Normally, the operation requires transferring a buffer to some output device. When the IOP has completed honoring the request, it places a completion code in the mailbox and sets INT for the Stack-machine requesting the I/O. This interpreter must halt execution and check mailbox to see if the I/O was performed successfully, completing the handshaking process. This protocol permits both Stack-machines to perform input/output independently of each other, provided both maintain strict memory boundaries.

Another function of the IOP is interfacing character code formats between the peripherals and the other two D-machines. When the machines are configured as Stack-machines, characters are passed to the IOP in 6-bit BCL (Burroughs Common Language). The IOP must convert this character set to ASCII for output to the line printer and CRT. A similar translation must be made for input data converting from either EBCDIC or ASCII depending on whether the input source is the card reader or the CRT.

3. Memory Interface

Since all three D-machines must share the 64K of main memory, a priority scheme was developed to resolve

memory reference conflicts. The main memory module is actually a single-ported, 32-bit word, core memory which can be made to appear multiported using a switch interlock unit (SWI) developed by Burroughs. The switch interlock controls the main data bus of the system, and resolves conflicts using a priority scheme. The D-machine with the highest priority is the IOP, with the priority of the other two machines being relative to their physical proximity to the IOP. Once a memory reference has been made, a D-machine may continue execution without waiting for a completion signal from the switch interlock. Although this technique of memory referencing minimizes unnecessary delay, it restricts the program from changing the read or write addresses or the content of MIR (write only) prior to a completion signal.

C. MICROINSTRUCTION TIMING

The Burroughs D-machine initiates a microinstruction once every clock cycle. The D-machines utilized for the AN/UYK-20 emulation operated from a one MHz internal clock, which produced a clock pulse once every microsecond. A D-machine designed with an eight MHz clock, emitter-coupled logic (ECL), and a faster memory cycle time, however, could execute eight times faster. This implies that advances in circuit technology can permit emulations to achieve improved speed and performance with no change in the microprograms.

Every microinstruction is executed using one or more sequential time periods, called phase 1, phase 2, and phase

3. A phase is a constant interval of time equivalent to one clock duration measured from the trailing edge of each successive clock pulse. Some microinstructions only require phase 1 to complete execution. Some require phase 1 and phase 3, and still others require phases 1,2, and 3. A new microinstruction is initiated at each clock cycle, allowing for overlapping of microinstruction execution in phase 1 and phase 3.

Microinstructions consist of two types. In a type 1 microinstruction, events can take place in all three phases:

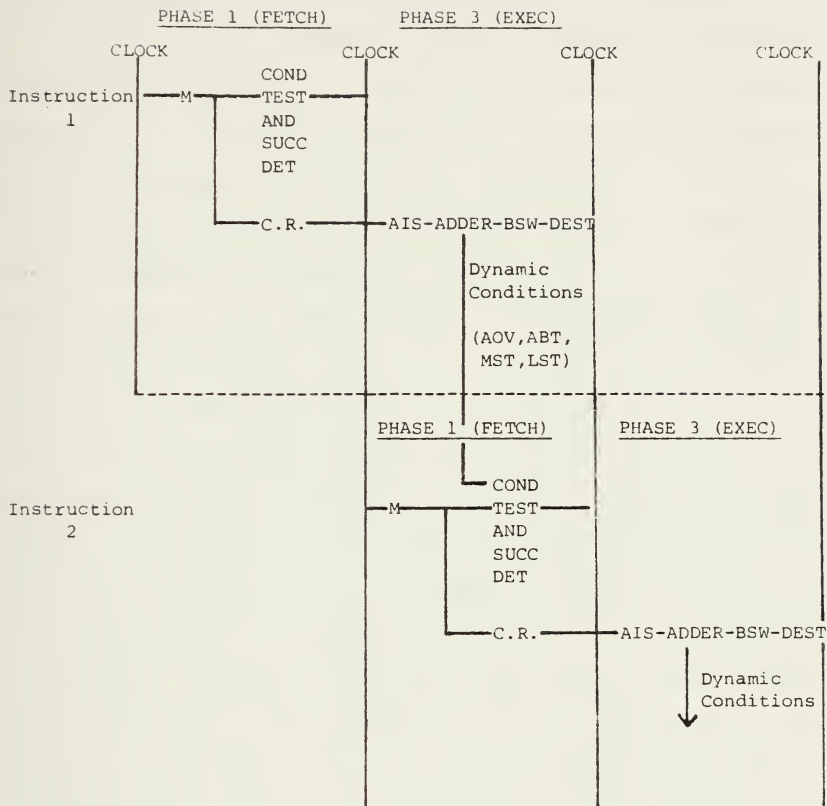
Phase 1: condition testing, (conditional) external operations or (conditional) logic operation initiation after completion of a prior logic operation, and successor M-memory address control.

Phase 2: a holding phase for phase 3 logic operation controls.

Phase 3: the completion phase for logic unit operations and destination register gating specified by logic operation.

During the optional phase 2 period, a type 1 microinstruction execution completion is held in abeyance while a subsequent type 2 instruction is executed. A type 2 microinstruction requires only phase 1 to complete execution, and involves literal assignments to three registers: LIT, SAR, and AMPCR. Phase 2 may also be initiated if the

next sequential type 1 instruction does not execute its conditional logic operation and therefore can complete its execution in phase 1 (Figure IV-3). Appendix D of Ref. 6 has a complete discussion of microinstruction timing.



M - MPM ACCESS TIME
 COND TEST AND SUCC DET - CONDITION TEST AND SUCCESSOR DETERMINATION
 BSW - BARREL SWITCH
 DEST - BARREL SWITCH OUTPUT DESTINATIONS; I.E., REGISTERS (B,CTR,ETC.)
 AND THEIR INPUT LOGIC
 C.R. - COMMAND REGISTER AND ASSOCIATED LOGIC
 AIS - ADDER INPUT SELECTION FROM COMMAND REGISTER

Timing Analysis, Type I Instructions

Figure IV-3 [17]

D. TRANSLANG

Microprogramming on the D-machine is accomplished using a microtranslator/assembler called TRANSLANG. TRANSLANG allows the programmer to write microinstructions mnemonically without concentrating on the bit patterns that compose the microinstructions themselves. TRANSLANG is written in ALGOL, the language of the B6700 Stack-machine. Nearly the entire language of TRANSLANG is composed of reserved words recognized by the ALGOL program. Each reserved word has a special meaning which causes the translator to construct particular microinstructions. A TRANSLANG instruction is equivalent to one microinstruction consisting of the set of parallel D-machine functions performed during the clock phases. TRANSLANG is a free form language and instructions may be written in almost any order. Multiple instructions may appear on a line, separated by a period '.'. TRANSLANG constructs include iterative mechanisms, input/output, assignment functions, control transfers, and Boolean, and computational operations. In addition, TRANSLANG permits label definitions and symbolic references for program control flow. Reference 6 is the programming manual for TRANSLANG and contains the complete syntax for the language. Appendix A of Ref. 10 documents additions to the TRANSLANG instruction repertoire.

A microprogrammer may construct complicated microinstructions that perform many different tasks, some interacting closely with D-machine clock timing. Microinstruction

gating to several devices permits a single TRANSLANG instruction to accomplish some or all of the following actions:

- a. test a condition.
- b. set/reset a condition.
- c. initiate an external operation.
- d. add.
- e. shift the result of an add.
- f. store the result into several registers.
- g. increment a counter.
- h. complement a shift amount.
- i. determine the successor microinstruction.

By judiciously composing his microprogram, a programmer may minimize execution time by taking advantage of microinstruction phase overlap and using highly parallel microcode.

The TRANSLANG assembler constructs an object program consisting of non-relocatable 56-bit microinstructions. TRANSLANG maintains a cross reference table that resolves label references during assembly. The object code created is stored on a disk and may be loaded into the micromemory of a D-machine using a special control word recognized by the operating system. Once loaded, the D-machine assumes the control structure dictated by the users microprogram.

V. AN/UYK-20 EMULATOR

A. EMULATION DESIGN

1. Functional Components

The architecture of the AN/UYK-20 emulator microprogram was developed using general guidelines provided by references and previous emulation experience on Burroughs D-machines [10]. The first decision incorporated in the emulator design was to integrate the entire emulation within one D-machine. Since the D-machines had the capacity to handle nearly 8K of microinstructions, no microprogramming capacity limitations were envisioned. The design objectives of a modularized, well-documented, structured microprogram could also be realized.

With the emulator entirely contained in one D-machine, several secondary benefits also existed. First, the emulator was more immune to hardware problems. If one D-machine was malfunctioning, the emulator could still be run on the alternate D-machine. Second, it was possible to have two AN/UYK-20 emulators resident in the system at one time. Not only would two emulators speed up testing of the individual emulation, but they would also permit their eventual use in a system configured for AN/UYK-20 multiprocessing. The third and final benefit of a totally integrated

emulator was recognized during the design of the input/output controller (IOC). Since the AN/UYK-20's IOC was capable of independent processing, an emulation of its IOC would not be possible within the same D-machine. An emulation of the IOC could be accomplished in a second D-machine, which would behave as an independent channel for the D-machine configured as the AN/UYK-20 processor. Although this emulation does not attempt to emulate the AN/UYK-20 IOC, the fact that a second D-machine exists makes its implementation a realistic extension to the project.

The emulator program organization was created following the basic guidelines of Ref. 17. The loader occupied the lowest section of M-memory with the emulator microcode following sequentially. Microprogram control passed from the loader to the emulator via the execution of a 'G' card which signified execution commencement.

The emulator microprogram was organized into three modules positioned such that forward address referencing would be minimized in the TRANSLANG assembler to save time and space. The utilities section was in the lowest portion of the emulator, since its routines were referenced frequently by the succeeding sections. Individual subroutines within the utility section were organized alphabetically.

The instruction and memory fetch routines comprised the next module. These routines incorporated all fetch options available in the AN/UYK-20 emulator.

The opcode routines occupied the last section of the emulator. The opcodes were arranged numerically with further indexing determined by the remaining fields of the individual instruction. Figure V-1 shows the M-memory mapping of the AN/UYK-20 emulator layout. Appendix F contains the emulation program listing.

2. Main Memory Organization

Of primary importance in the emulation design was the logical organization of the Burroughs system's main memory. Since the AN/UYK-20 was a 16-bit word machine, it was decided that only the lower half-word of a Burroughs 32-bit word would be used by the emulator. This design restriction permitted the addressing structure of the AN/UYK-20 to be directly projected on the D-machine. A one-to-one correspondence between the memory address of the AN/UYK-20 and the Burroughs system was also achieved.

Since the number of high-speed registers available in the D-machine was small, a 1K portion of main memory was reserved for the AN/UYK-20 addressable register set, the page address registers, the temporary storage space, and the emulation buffers. The mapping of the AN/UYK-20 high-speed registers to main memory greatly simplified the microprogramming requirements of the emulation, but added considerable execution time overhead. Memory access times for the mapped registers were much slower than the actual transfer

0000

LOADER
UTILITIES

INPUT/OUTPUT CONTROLLER
FETCH
OPCODE ANALYSIS
OPCODE 00
OPCODE 01
OPCODE 02
OPCODE 03
• •
• •
• •
OPCODE 74
OPCODE 75
OPCODE 76

AN/UYK-20 EMULATION ORGANIZATION

Figure V-1

rates of the AN/UYK-20 registers. Figure V-2 shows the complete main memory mapping of the emulation reserved storage area.

3. Emulation Program Status Word

Although the emulation duplicated all the AN/UYK-20 registers, the registers which comprised the program status word (PSW) possessed unique characteristics. Status register 1 was combined with the program address register to form a single 32-bit PSW. The emulator's PSW always co-existed in the A1 register of the D-machine and in main memory during execution of AN/UYK-20 programs. The fields of SR1 were modified to include certain emulator toggles, along with the normal condition bits (Figure V-3). The condition bits for DMA and non-destructive read only (NDRO) mode were removed from the SR1 since they were hardware features of the AN/UYK-20 that would not be emulated.

Status register 2, the remaining 16 bits of the AN-UYK-20's PSW, was not resident in any D-machine registers during emulation execution for two reasons: the emulation could not afford the luxury of 48 bits of reserved register space, and SR2 was less frequently referenced than SR1 and the P-register. Consequently, SR2 had to be read from its reserved location in main memory. The contents of the upper 16 bits of SR2's memory location also contained additional emulation toggles which were used by the debugger package (Figure V-3).

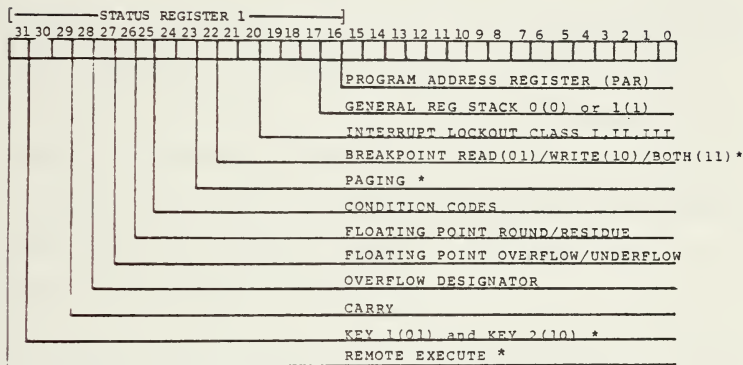
MAIN MEMORY MAPPINGS

DECIMAL ADDRESS	OCTAL ADDRESS	USE

0 - 15	0 - 17	GENERAL REGISTER STACK 1
16 - 31	20 - 37	GENERAL REGISTER STACK 2
32	40	PROGRAM STATUS WORD
33	41	BREAKPOINT REGISTER
34	42	STATUS REGISTER 2 (SR2)
35	43	NEXT LOAD ADDRESS
36	44	CLOCKTIME
37	45	REAL TIME CLOCK
38 - 43	46 - 53	WORKSPACE (TEMP STORAGE)
44 - 49	54 - 61	STACK (TEMP STORAGE STACK)
50 - 53	62 - 65	I/O COMMAND WORDS (IOCW)
54 - 119	66 - 167	UNUSED
120 - 121	170 - 171	HEX ADDRESS FOR INPUT
122 - 141	172 - 215	INPUT CARD BUFFER
142 - 152	216 - 230	UNUSED
153 - 185	231 - 271	OUTPUT PRINT BUFFER
186 - 205	272 - 315	CRT BUFFER
206 - 229	316 - 345	ERRORLIST
230 - 767	346 - 1377	UNUSED
768 - 1023	1400 - 1777	PAGE ADDRESS REGISTERS

Figure V-2

ACTIVE PSW: RESIDES IN LU REGISTER A1 (MEMORY ADDRESS 32)



[— STATUS REGISTER 2 —]

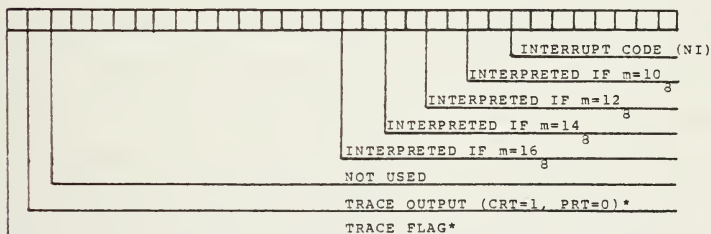


Figure V-3

4. D-Machine Registers

with only a limited number of high-speed registers available in the D-machine, the emulation design had to include a well-developed plan for their usage. Since only seven registers in the D-machine were 16 bits or longer, they were used exclusively for manipulating the AN/UYK-20 addresses, instructions, and data. The register environment had to be consistent throughout every execution cycle to allow utility routines to be called in the same manner by all opcode subroutines. The primary goal of the emulation was to use as few memory references as possible to conserve execution time and memory space. Judicious use of the existing registers was a necessity.

Several factors had to be considered before selecting a register for a particular function. The most important consideration was its flexibility within the D-machine. The B register, for example, was used as a general purpose register, since its contents could be gated through a masking filter prior to being utilized. A second factor was the type of operand it could contain. Double-word operands (32-bit) could only be stored in the 32-bit logic unit registers while 16-bit operands could also be stored in the memory control unit registers.

The A1 register contained the emulation's 32-bit program status word (PSW) at the commencement of the emulator program. It was not affected by individual instruction

microcode, except when incrementing the PAR, or when a particular instruction modified the PSW. Emulator toggles resident in SR1 could not be altered by an AN/UYK-20 instruction because their settings were independent of program execution.

The A3 register held the instruction word for the duration of its execution cycle. Each field of the instruction could be decoded and interpreted from the A3 register without having to retrieve it from memory. Once the instruction had been completely decoded, A3 was made available as a scratch pad register.

The A2 and B registers were used as scratch pad registers during each opcode execution cycle. In general, their contents were volatile, except when they were specifically documented in the the program listing. The contents of A2, for example, was not altered in the EMULIN subroutine, because of its use in the calling fetch routine. A2 and B were manipulated as either single or double-word operands during the arithmetic operations.

The only remaining logic unit register was the memory information register (MIR). MIR was used for storing information into memory and as a temporary storage location. Intermediate results were deposited in MIR during instruction execution and returned through the B register.

The base registers, BR1 and BR2, were used as storage for addresses and single-length operands, and for

temporary storage of intermediate results. In addition, all memory addressing in the emulation was accomplished using the lower 8 bits of BR2 and MAR (MAR2). These memory control unit registers had to be used carefully, because they required a sequence of several microinstructions to properly reference their contents.

B. LOADER

The loader incorporated into the AN/UYK-20 emulation provided a simple mechanism for loading AN/UYK-20 instructions into main memory (S-memory) of the Burroughs microprogramming system (Figure V-4). Its control word repertoire was flexible, allowing a variety of AN/UYK-20 program environments. Job control statements were included to execute and halt individual programs anywhere in S-memory.

The loader module consisted essentially of a scanner and a translator written in the microcode. Information was read into a 20-word buffer from cards or CRT input, then the buffer contents were scanned for control code consisting of one or more characters. Once these characters were interpreted, control was passed to the translator section which decoded the rest of the data in the buffer and performed the required function. The translator section consisted of a variety of routines that handled specific control words in the loader repertoire. The loader control statements, however, had to appear in a logical sequence (See Appendix A). All loader control statements are contained in Appendix B.

AN/UYK-20 LOADER

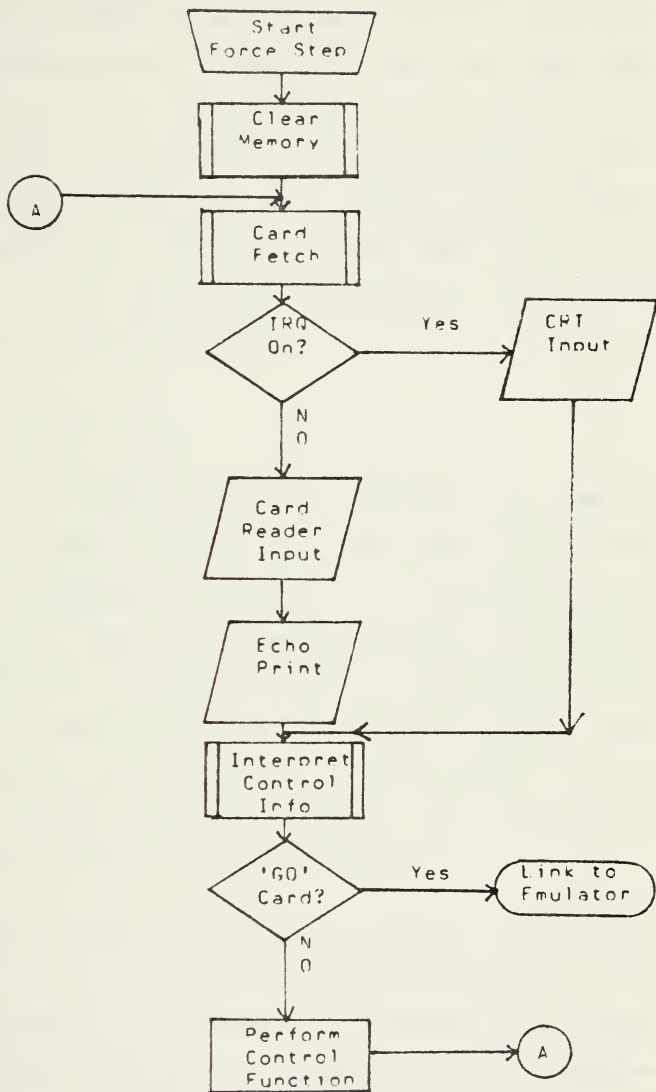


Figure V-4

The actual interface pipeline between the loader and the emulator consisted of two small emulator utility routines which started and stopped the external clock of the Burroughs microprogramming system. These routines were inserted for emulation timing purposes, and provided a 'stop watch' for AN/UYK-20 programs. The time recorded (in milliseconds) by this 'stop watch' was placed in a reserved memory location in the emulation memory map, and could be read using either a trace control instruction ('T'), or a machine status control word ('M').

C. THE FETCH MODULE

In order to emulate AN/UYK-20 execution and memory referencing, fetch microcode was developed which incorporated memory addressing algorithms and instruction fetch routines. Since both data and instructions were equally accessible from the processor, the memory addressing scheme was closely linked to the instruction fetch concept. Data and instructions could be interspersed throughout memory, and proper program execution required that the program address register point to an instruction word.

The emulation used two routines for memory addressing, EMULIN for reading, and EMULOUT for writing (Figures V-5, V-6). These subroutines performed both paging and break-point checking, depending on toggles set in the program status word. Paging was incorporated into the emulation in order to gauge the execution overhead required in emulating

the AN/UYK-20's paging scheme. The paging scheme implemented in the emulator divided main memory into 256-word pages, instead of 1024-word pages used by the AN/UYK-20. Since the Burroughs D-machine was organized for 256-word pages, the microprogramming required for the paging was straightforward. The 256 page address registers resided in the emulator's memory mapping, each initialized to the page number corresponding to their relative address (0-255). The paging algorithm imitated the AN/UYK-20's method of page addressing, and included the setting of a page modification bit.

The breakpoint option was added to provide a method of debugging AN/UYK-20 programs, once the emulation was completed. EMULIN and EMULOUT tested toggles set in the PSW to determine if breakpoint read, write, or both was desired.

The memory addressing convention required all memory references from 1K to 64K to use EMULIN and EMULOUT. All memory references to the memory mapping area (0-1023) did not use these routines, but instead utilized absolute memory referencing microcode.

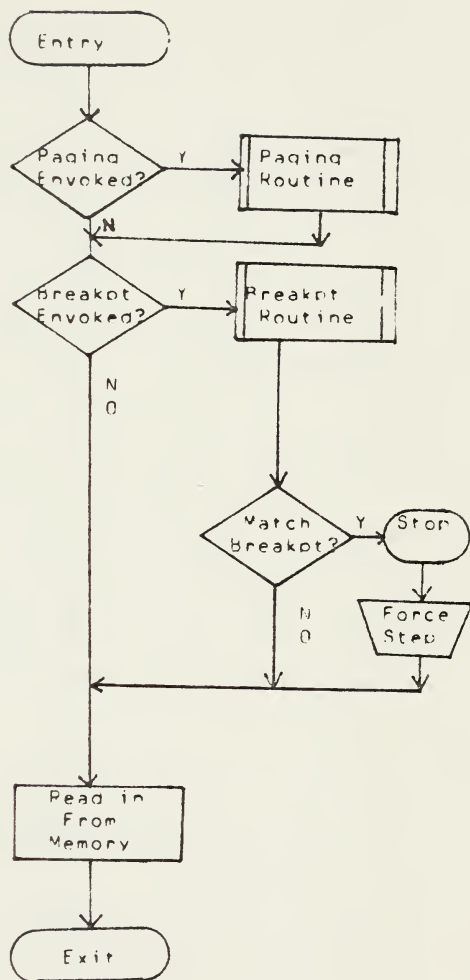


Figure V-5

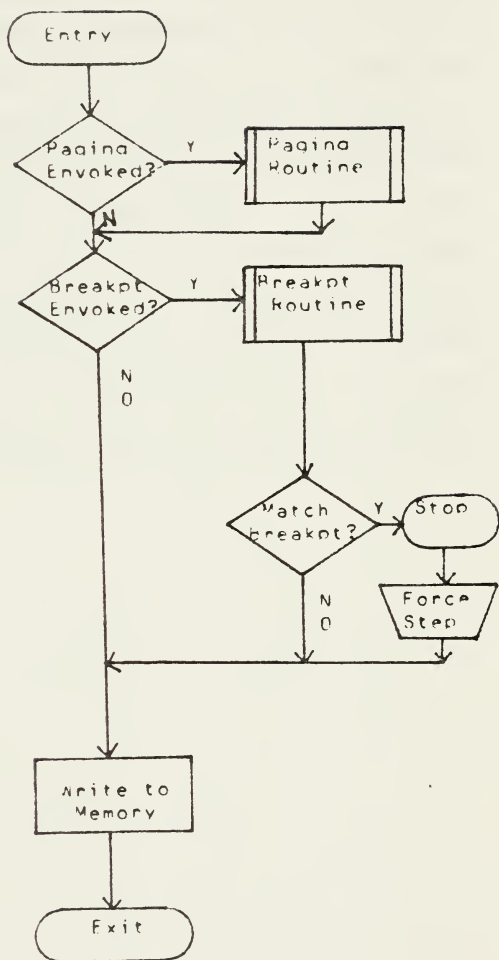


Figure V-6

The instruction fetch routine (IFETCH) retrieved instructions from main memory based on the contents of the program address register. Since IFETCH used EMULIN to read the instructions from memory, it could operate in a paging environment, with or without breakpoints (Figure V-7). IFETCH was also responsible for incrementing the PAR, and for testing the trace toggle prior to fetching an instruction. IFETCH was capable of retrieving any instruction word in the program address space (1024-65,535). In the event that the upper memory limit was reached, IFETCH would set the PAR to 1024 and continue execution. Trace toggle testing was inserted into IFETCH as part of the built-in debugging package. Since IFETCH was called prior to every instruction, it was the logical choice for placing a call to the debugger.

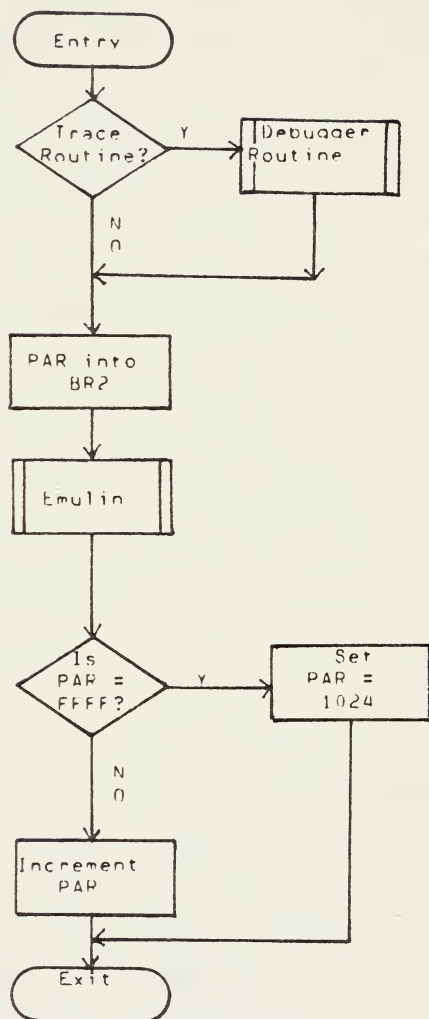


Figure V-7

D. OPCODE IMPLEMENTATION

All of the 205 individual instructions emulated were microprogrammed using an identical instruction decoding mechanism. The routines that performed the required operations were terminal nodes on a large tree network, whose root was the OPCODE routine. OPCODE fetched an instruction and isolated the operation code field. The binary value of the opcode field was an index to an operation jump table containing individual opcode M-memory addresses. Within each opcode routine was microcode which isolated the sub-function 'f' field of the instruction and used its value as an index to the next level of opcode analysis. Depending on the opcode, this last jump could identify which instruction was to be performed. If it did not, further analysis of the 'm' or 'a' fields provided the final index to the instruction. Instruction formats are described in Appendix C.

After the instruction had been executed, control passed back to the opcode routine, and then the execution cycle was repeated for the emulation of the next AN/UYK-20 instruction. The AN/UYK-20 programmer could cause this microprogram loop to be exited by either inserting an executive return instruction (03,0,a,00) which caused a 'priority interrupt' and halted program execution, or by coding an instruction that was not implemented, not assigned, or caused a division overflow. The last three cases caused an execution fault, while the former resulted in normal program termination (Figure V-8).

AN/DYK-20 EMULATION GENERAL STRUCTURE

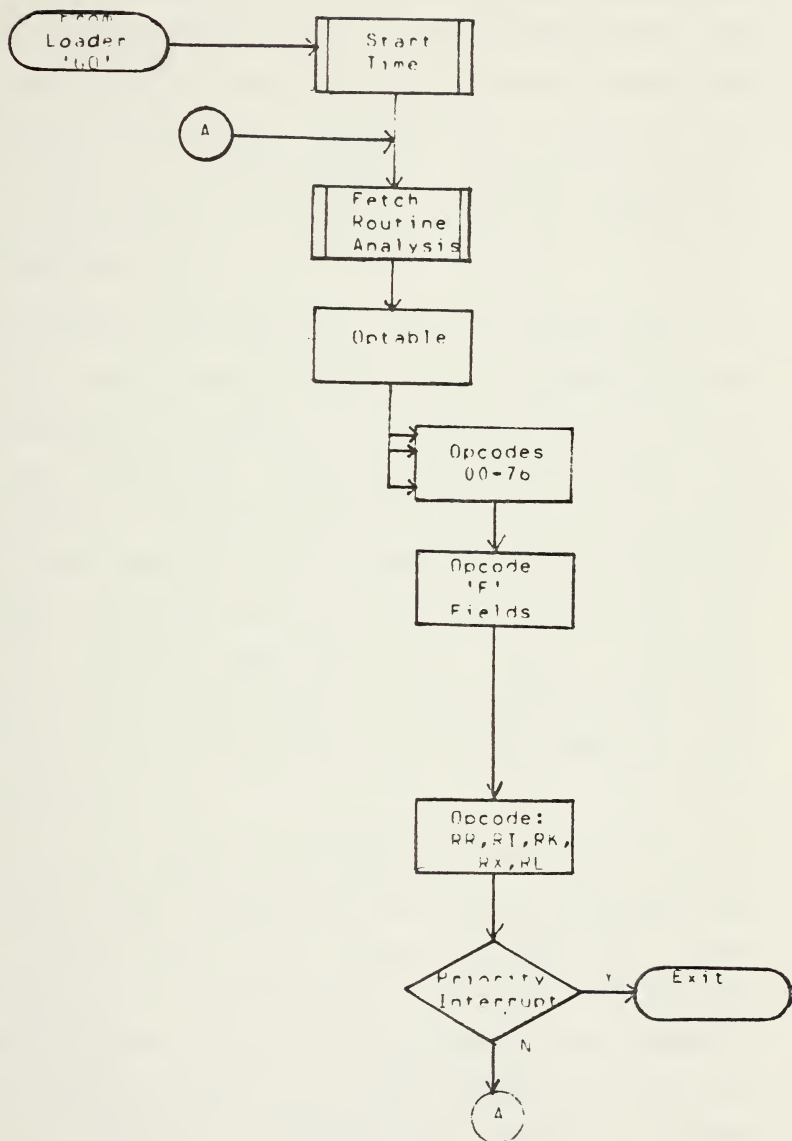


Figure V-8

The instruction fetch mode of OPCODE assumed that the PAR always pointed to an instruction word. Double-word instructions always had their 'y' field fetched after they were determined to be two words in length. Conditional double-word instructions performed their condition test before the 'y' field was fetched. If the test failed, the PAR was incremented by two prior to returning to OPCODE for continued execution.

The OPCODE routine was also written to accommodate the AN/UYK-20 'remote execute' instruction. When this operation was performed, a bit was set in the PSW which would indicate that one instruction out-of-line was being executed. For this operation, the current PSW was stored in memory, and the PAR was loaded with the address of the instruction to be executed. The OPCODE routine always checked the remote execute bit during an instruction cycle. If the bit was set, it would fetch the instruction indicated by the remote execute PAR, and restore the actual PAR, incremented by two, into the PSW.

Some of the opcode repertoire of the AN/UYK-20 was not emulated. Those instructions that were not emulated, however, retained slots in the opcode hierarchy for future inclusion. Any instruction not implemented by the emulator caused the machine to fault, and printed an error diagnostic on the selected output device ('NOT IMPLEMENTED - EXECUTION ENDS '). Similarly, locations were reserved for those instructions not assigned by the AN/UYK-20 were reserved

locations in the emulation . This permitted the emulator to be responsive to any future AN/UYK-20 hardware modifications. Whenever an instruction to be executed was not assigned, the emulator generated a fault interrupt and printed an error diagnostic on the selected output device ('FAULT INTERRUPT - EXECUTION ENDS').

E. UTILITIES

Although each emulated instruction performed different operations, each depended upon a common set of utility subroutines to accomplish their task. These subroutines varied in complexity, but each performed a function that contributed to successful instruction execution. A simple operation often required register addressing, condition code setting, and memory addressing, before the task was completed.

Utility subroutines were included in the emulation whenever feasible to simplify microprogramming and to alleviate programming redundancy. These subroutines were called using the successor command constructs available in TRANSLANG. Depending on the purpose of the utility routine, parameters were passed via D-machine register(s) or condition bit(s). This information was utilized by the utility in determining what operation was to be performed. In the carry subroutine, for example, a local condition bit was passed which indicated the appropriate condition code to be inserted into the PSW. A more complex example, the RX

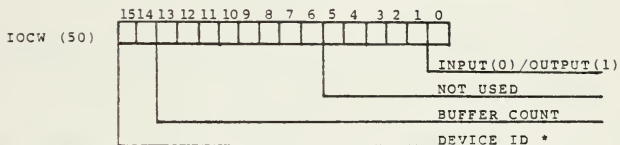
format utility routine, required two parameters to be set by the instruction. Register A3 contained the instruction word and LC2 was set or cleared depending on whether or not byte formatting was required. The RX routine called other routines and could perform considerable processing before the final result, the effective operand address, was returned to the calling routine via the B register.

The utility section encompassed a number of routines which performed complex data manipulation. Arithmetic utilities for multiplication, division, and square root were accessed by nine separate AN/UYK-20 instructions. Indirection routines, which were called by the RX format routines, emulated the AN/UYK-20 cascaded addressing capabilities. A general purpose move subroutine permitted up to 256 cells of main memory to be moved from one location to another. This routine was used by the emulator's error diagnostic utilities, as well as the load and store multiple address register instructions.

F. INPUT/OUTPUT CONTROLLER

Although the AN/UYK-20's IOC was not emulated, several of its design features were imitated in creating a general purpose input/output controller for the emulator. The input/output command instruction (35 RR) initiated the I/O sequence, and reserved several cells in the memory mapping for I/O control words (Figure V-9). These control words contained fields which indicated what peripheral device was

AN/UYK-20 EMULATION
INPUT/OUTPUT COMMAND WORDS



BUFFER ADDRESS
THE MICROCODE ALTERS THE IOCW + 1
DURING BUFFER TRANSFER



BUFFER SIZE
USED FOR CRT OUTPUT



NUMBER OF SECTIONS
USED FOR DISK TRANSFERS (NOT IMPLEMENTED)
* CODE: 00 - CRT
01 - PRT (OUTPUT)/CRD RDR (INPUT)
10 - DISK (NOT IMPLEMENTED)

Figure V-9

selected, the number of buffers that would be passed, whether input or output was desired, and where the buffer was located. Two additional words were reserved for control data required to interface with the disk and the CRT. The disk input/output microcode was not incorporated into the controller, but the framework was provided so the IOC could be readily inserted in the future.

The emulator's IOC section was located in the utility section of the emulation, and operated asynchronously with the Burroughs IOP. Since the IOC was collocated with the emulator in the same D-machine, it was not capable of independent processing. Once it was initiated, emulation execution waited until input/output was completed. The emulation had to use the Burroughs Common Language, since the IOP was microprogrammed to accept only this character set from other D-machines.

In order to transfer a 33-word line printer buffer to the IOP, the AN/UYK-20 emulator IOC had to use 66 AN/UYK-20 words. In order to send 20 words to the CRT, the IOC had to construct a 40-word buffer. The COMPRES subroutine packed an AN/UYK-20 buffer of 66, 16-bit words into a 32-bit, 33-word buffer, suitable for output to the IOP.

Similarly, on input, a 20-word IOP buffer from the card reader or CRT expanded into a 40-word AN/UYK-20 buffer. In this case, the EXPAND subroutine split every packed 32-bit word of the IOP's buffer into two 16-bit words, suitable for processing by the emulator. These additional data transformations were required because of the buffering requirements of the Burroughs IOP.

VI. EMULATION TESTING

A. METHOD OF TESTING

The fundamental testing technique utilized throughout the development of the emulation consisted of three distinct phases: 1) each module was independently assembled and debugged until successful assembly was achieved, 2) each program segment was then tested for accuracy of the intended operation, and 3) the composite emulation program was tested under actual program conditions allowing for interaction among several modules. A debugging package, which was developed early in the emulation project, was the primary test vehicle for verifying emulation coding. Development and testing of the loader was, however, completed prior to the creation of the debugger.

Loader testing was accomplished by monitoring the control panel lights and synthetically halting program execution via the insertion of 'WAIT' statements, forcing the desired register to be transferred to 'MIR', and then re-examining the panel lights. This process was extremely tedious and time consuming but proved to be an invaluable tool for recognizing peculiar TRANSLANG constructs and microinstruction execution side effects.

Each module was subjected to an extensive desk checking process in order to reduce trivial assembly errors before running it on the machine. After a successful assembly had been achieved, the code was merged with previously existing assembled modules. The emulator was expanded as more modules were added, with utility routines being incorporated prior to the opcode programs.

Initially, the loader was designed, programmed, and thoroughly tested prior to the microprogramming of any AN/UYK-20 instructions or utilities. With the loader implemented, the emulation of the UYK-20 instruction repertoire could proceed with a minimum of loader induced problems.

Independent opcodes and various utility routines were added to the previously assembled programs. The final emulation consisted of a total of 205 opcodes, a set of utility programs, and a workable loader.

In the next phase of testing, every module was subjected to a representative number of test cases which demonstrated how closely they compared to the documented AN/UYK-20 operation. Artificial environments were created to subject every opcode to a variety of situations. Whenever the operation of the opcode disagreed with the documented AN/UYK-20 operation, the opcode's function was thoroughly researched. The opcode's side effects were categorized and questions formulated. Often the answers could only be obtained from the Univac field engineer.

To illustrate the complexity of testing, an add instruction was tested with numerous operand combinations: two positive operands, two negative operands, one of each sign yielding a negative result, opposite signs producing a positive result, and opposite signs producing a zero sum. During each addition operation, the overflow, carry, and condition bits had to be monitored in status register 1 to verify their appropriate setting. This level of detail was achieved with all of the implemented opcodes in order to produce an efficient and accurate emulation.

Throughout the entire testing scenario, which composed 20% of the emulation project, the debugger routine (DUMPREG) provided the necessary information for examining opcode execution. A representative sample debugger output is provided in Appendix D.

DUMPREG permitted a snapshot of both AN/UYK-20 general register stacks, PSW, SR1, and SR2, in addition to the D-machine registers (A1, A2, A3, BR1, AMPCR, MIR) and the Burroughs external clock. DUMPREG possessed sufficient flexibility to be incorporated into the microcode at any point. In the final emulator, however, the debugger is user specified, and will dump the AN/UYK-20 emulator environment either at every instruction fetch and program stop, or when called by a loader control card.

B. SAMPLE TEST PROGRAMS

After subjecting the entire emulation to extensive testing, some representative test programs were developed to demonstrate the feasibility of the emulation and its capabilities and performance as compared to an actual AN/UYK-20. There were two programs which were selected because they incorporated numerous emulation features. The two programs were the solution of simultaneous linear equations by Cramer's rule and generation of prime numbers. It must be noted that streamlined program design was not emphasized but rather utilization of a variety of opcodes and features of the emulation.

The program for solving linear equations contained a total of 28 opcodes, requiring 28 opcode execution cycles and 43 instruction fetches. The program demonstrated all four fundamental mathematical operations, numerous store and load functions, a comparison test and a jump instruction. The capability of performing card reader input and output was added when the emulator IOC was completed.

The prime number program demonstrated 30 instructions which required 116 opcode execution cycles, and 122 instruction fetches. This program illustrated numerous comparison tests, looping structures, several jump instructions, a load multiple instruction, addition, and division. The test programs are included in Appendix E.

C. TEST RESULTS

The performance analysis of the test programs consisted primarily of running numerous test cases, examining the results for accuracy and computing the total time required to execute the emulated AN/UYK-20 programs. The timing of the programs was accomplished by using an external clock available on the ICP. The clock time provided a fairly close representation of the emulation execution time, but it cannot be considered a completely accurate measure since the emulation must interrogate the IOP to retrieve the external clock contents. Approximately 50 microseconds are used when sampling the clock.

The time requirements of the AN/UYK-20 program execution were hand-calculated by summing the published instruction execution times as presented in Ref. 21. The emulation performance ratio (EPR) was computed merely to give an approximate indication of the emulation performance. The EPR is the ratio of measured Burroughs emulation time to the calculated AN/UYK-20 execution time. Two EPR figures are recorded: one with paging implemented and one without. The paging EPR figure is significant only in that it indicates how much additional overhead must be incurred when emulating the paging mechanism of the AN/UYK-20. The required additional execution time was about 26%. Naturally, paging overhead is directly proportional to the number of instruction fetches or memory references performed during a program.

The results of program testing are as follows:

	CRAMER'S RULE	PRIME NUMBERS

Number of Opcodes	28	30
Number of Fetches	43	122
Execution Cycles	28	116
Time w/o paging	5000 usec	13000 usec
Time with Paging	8000 usec	17000 usec
AN/UYK-20 Time	78 usec	193 usec
EPR w/o Paging	64 :: 1	67 :: 1
EPR with Paging	103 :: 1	88 :: 1

These figures represent only approximate comparisons of the two machines. These computations provide an estimate of emulation characteristics. An effective EPR without paging was projected to be 65::1.

VII. SUMMARY AND RECOMMENDATIONS

A. EXPERIENCE WITH HARDWARE

The emulation project provided the unique opportunity of learning about two computer systems. The Burroughs D-machine demanded a detailed knowledge of hardware operation, as well as a thorough understanding of the microprogramming language, TRANSLANG. The computer architecture and processor capabilities of the AN/UYK-20 had to be investigated and then integrated into the control store memory of the Burrough's D-machine.

On several occasions, hardware malfunctions with the Burroughs equipment prevented normal system operation. The card reader was inoperable for several weeks, and the disk drive unit had to be repaired several times. During the final three weeks of project development, one D-machine ceased to function properly. This restricted emulation testing to the remaining D-machine.

Although these hardware difficulties impeded normal progress of the project, they did not prevent the emulation from successfully being completed. If a hardware problem prevented emulation testing or debugging, other modules were designed and testing was postponed. This permitted continual emulation development regardless of the hardware

status. In addition, considerable time and effort was invested in trouble-shooting hardware malfunctions, so they could be isolated, diagnosed, and repaired.

B. LESSONS LEARNED

The emulation project brought together many different computer science techniques and disciplines which will be useful in future computer science endeavors. A great deal of experience in both computer architecture and operation was gained in two different types of computers. Microprogramming provided new insight into computer design and revealed many potential applications for programmable control store machines.

Since emulations normally require a large development effort, this project had to incorporate judicious system design and project management principles in order for it to be completed successfully. Careful monitoring of critical stages of the emulation, and coordination of the programming effort to meet scheduled requirements, was necessary throughout this research. The small programming team concept proved to work extremely well in this project.

Finally, several software practices were strictly followed that proved to be invaluable in constructing the emulation. First, modular programming succeeded in partitioning the emulation into discrete modules which could be designed, coded, tested, and implemented individually. The

emulation was constructed in segments, using the previously verified modules as a test bed for the new modules being built. Structured programming and prolific documentation were useful in developing each microprogram routine because they permitted each team member to understand the function of the program and how it could be used.

C. EMULATION PROBLEMS

The most significant problem of the emulation was not having had any experience with an AN/UYK-20 and not having anyone readily available with prior AN/UYK-20 experience. This lack of knowledge created some anxiety when attempting to analyze the ramifications of individual opcodes.

The idiosyncrasies of certain opcodes were not made self-evident by the AN/UYK-20 software manuals. Consequently, numerous code modules were redesigned when more detailed information was provided by a UNIVAC field engineer. This proved to be very time consuming, inefficient, and frustrating.

Another emulation difficulty was the lack of working registers in the host machine as compared with the target machine. While the AN/UYK-20 has either 16 or 32 general registers, depending on whether the second stack option is incorporated, the D-machine has effectively only seven workable registers containing 16 bits or more. Therefore, all AN/UYK-20 registers had to be mapped into S-memory which

created much longer register read/write times. This created significant register manipulation problems which in some cases required main memory references for instructions intended to be strictly register-to-register operations. Consequently, increased execution times resulted in a higher emulation performance ratio (EPR), decreasing the overall emulation performance.

D. RESULTS

Emulating the AN/UYK-20 on a Burroughs D-machine required a considerable amount of preparation and planning before any results were realized. An in depth analysis of each computer's architecture and operating characteristics was conducted to insure that an emulation was feasible in the allotted time period. From the inception of the project, the goal of the emulation was to implement a standard AN/UYK-20 processor. This decision was based on the capability of the D-machine and an estimate of how much time would be involved in developing, debugging, and testing the final product. Although this goal was achieved, it was felt more time could have been devoted to testing and verifying emulation operation.

The AN/UYK-20 emulation was a highly complex microprogramming project involving numerous data structures and transfer protocols. A total of 205 AN/UYK-20 instructions were emulated out of nearly 290 instructions in the repertoire (including all IOC, math op, and clock interrupt

opcodes). AN/UYK-20 diagnostic programs and user programs will establish the validity of the emulator's construction.

E. RECOMMENDATIONS AND FOLLOW-ON TOPICS

Although 205 opcodes have been implemented, tested, and developed into a working emulation, there are still many challenging avenues to pursue in creating the complete emulation package. First, testing is a continuous process and should be performed in conjunction with code optimization. A comprehensive code optimization effort could improve the EPR without sacrificing code readability.

Second, the floating point and 'math pac' options could be implemented. The addition of floating point arithmetic, trigonometric, and hyperbolic functions would significantly strengthen the scientific capabilities of the emulation. This would be an extremely strenuous undertaking but would permit more tactical data applications of the emulation, increasing its value to the Navy.

One area which could be examined is to perform a comprehensive timing analysis between an AN/UYK-20 and the emulation. This could consist of collecting numerous benchmark programs from Navy AN/UYK-20 installations, where performance data could be accurately obtained. These programs could then be run on the emulator, and analyzed with the benchmark results. The feasibility of replacing or substituting emulation host machines for target machines could be

addressed and supported by the timing analysis study.

An emulation of the AN/UYK-20 input/output controller could be incorporated into the emulation without serious difficulty. Since a second D-machine is available, the IOC channel processor instructions could be emulated and inserted into that D-machine's control memory. The independent processing characteristic of the AN/UYK-20 IOC would be fulfilled using this arrangement.

Finally, when the Burroughs system is linked with the computer science department's PDP 11/50, the AN/UYK-20 emulator could be connected to that system's peripheral resources. This would allow future incorporation of an AN/UYK-20 ULTRA assembler and a CMS-2M compiler into the PDP 11/50 system which could then produce machine language object code files for the AN/UYK-20 emulator to execute.

APPENDIX A. AN/UYK-20 EMULATOR USER'S MANUAL

This description is designed to provide sufficient information to operate the AN/UYK-20 emulation program. It is assumed that the informal Burroughs D-machine manual in the Burroughs laboratory will supply adequate power-on instructions and solve any hardware operating difficulties which may arise.

The procedure for utilizing the AN/UYK-20 emulator can be divided into four phases:

- 1) selection of the necessary loader control cards (JCL).
- 2) selection of the AN/UYK-20 program instruction set.
- 3) implementation of phases one and two into the required card or CRT format.
- 4) actual hardware implementation on the Burroughs D-machine resulting in program execution.

Phase one can be achieved by selecting the desired loader control cards described in Appendix B. The card format is identical to the CRT format, except that the CRT requires the user to <carriage return> at the end of every line of input data.

Phase two is accomplished by creating the AN/UYK-20 program from a subset of the 205 emulated instructions.

Phase three consists of keypunching the desired JCL and AN/UYK-20 program in the format illustrated in Appendix C.

The resulting job deck will typically be assembled as follows:

?SMLoad TED/UYK20-OBJECT % loads the emulator into
micromemory

L 00004 % load the program into
page 4

C 00206 FAULT INTERRUPT-EXECUTION ENDS "

C 00214 NOT IMPLEMENTED-EXECUTION ENDS "

C 00222 DIVIDE OVERFLOW - FAULT ENTERED"

other desired JCL

AN/UYK-20 Program

03,0,a,00 % priority interrupt
(a = 00-17 octal) (mandatory card)

G % commence program
execution

M (or M1) % machine status (reg. dump
at termination)

E % end job card

Finally, phase four consists of the entire program deck being loaded into the card reader. It is assumed that the IOP is at address 0015 hex, the selected interpreter is at address 0549 hex, the line printer is 'READY', and the IRQ switch is 'off'.

The IRQ switch is a three-way toggle switch mounted on the right side of the interpreter. In the up position, IRQ is 'on', horizontally it is 'off', and the downward position is used for external functions.

If either the interpreter or the IOP is not at the proper starting address, clear them by depressing the 'CLEAR' button on each unit. If this procedure does not remedy the situation, consult the user's manual in the laboratory.

Upon reading the ?SMLOAD card, the system will load the AN/UYK-20 emulator object program into the micromemory of the selected interpreter. The program address counter (on the interpreter) will be at the beginning of the emulation program, address 0000 hex. At this point, the user can select CRT input and output by placing the IRQ switch to the 'on' (upward) position. If CRT input is not desired, leave the IRQ switch in the 'off' (horizontal) position. Next, force step the interpreter by momentarily depressing the FST button (uppermost push button on side panel of the interpreter). Do not hold in the FST button. This may cause some undesirable side effects.

After depressing the FST button, the AN/UYK-20 program is loaded into S-memory. If the input is expected from the CRT, the user must enter his program from the console. Otherwise, the emulator will request cards from the card reader. Once the program has been loaded and the 'G' card read, program execution begins.

After successful program termination, the program returns to the loader and asks for more input. At this point, the user may terminate his job, or start another load sequence. It should be remembered that the AN/UYK-20 emulator has been designed for monoprogramming execution. It executes one program at a time, but can execute any number of programs in sequential order if desired. When the job is terminated, the D-machine returns to its starting address (0000 hex) and awaits further processing. When the user returns to the start address, the system is effectively 'master cleared' since S-memory will be cleared prior to executing any further jobs. It is not necessary, however, to use the ?SMLOAD card for additional programs or program re-runs because the emulator object code still resides in micromemory.

If the results of program execution are not as anticipated, use of either a 'T' or 'T1' option (trace card) is recommended to provide the user with a fetch-by-fetch program trace with the output to the printer or CRT respectively.

APPENDIX B. LOADER CONTROL CARD FORMATS

CARD COLUMNS

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	-	20

B	R																
	W																
	B																

C																	
D																	

E																	

G																	

I	1																
	2																

L																	

M																	
	1																

P																	

R																	

S	1																
	2																

T																	
	1																

Note: All numeric fields are in decimal.

LOADER CONTROL CARD DESCRIPTION

Control Card Identifier		Description
B	---	The 'B' card is used to implement the breakpoint feature of the emulation. This feature allows the user to specify a decimal address in columns 4-8. Column 2 must contain a R or W to breakpoint on a read or write operation respectively. The default condition is breakpoint on both read and write.
C	---	The 'C' card is used to insert character strings into memory. The character string starts in column 9 and continues until terminated by a quote symbol (") or the string reaches column 80. The decimal address where the character string will be written is given in columns 4-8. A blank or zero address field causes the character string to be inserted at the current load address.
D	---	The 'D' card is used to store decimal data from columns 16-20 into the memory address found in columns 4-8. A blank or zero address field causes the data to be inserted at the current load address.
E	---	The 'E' card is used to indicate the end-of-job and therefore is a mandatory control card for job separation.
G	---	The 'G' card or 'G0' card is used to start execution. The starting decimal address is in columns 4-8. The default value of a blank or zero address field will cause program execution to start address 01024.

- I --- The 'I' card or set index register card is used to store decimal data from columns 16-20 into the register designated in columns 7-8, into the general register stack (1 or 2) specified in column 2.
- L --- The 'L' or load card is used to partition memory into 256, 256-word (32-bit) pages and to load the program into the decimal page as referenced by columns 4-8. Pages 0-3 should not be used because they contain the required emulation register mapping and established workspace. The 'L' is a required control card, and it is recommended that it be the first JCL card.
- M --- The 'M' card or machine status card provides a register dump, wherever it is inserted. It is normally placed between the 'G' and 'E' cards in the JCL deck. If a 1 is placed in column 2, the machine status dump will be sent to the CRT. This dump will contain the current value of all AN/UYK-20 general registers, the PSW, SR2, next instruction address, the breakpoint address, and clocktime.
- P --- The 'P' card is used to implement paging.
- R --- The 'R' or reserve space card is used to reserve memory space as specified decimally in columns 4-8.
- S --- The 'S' card is used to simulate setting of the program stop switches (1 and 2) on the AN/UYK-20 maintenance panel. Column 2 must contain a 1 or a 2. Two cards are required if both switches are to be set.
- T --- The 'T' or trace card provides a fetch-by-fetch program trace, dumping the entire machine status on every fetch cycle. If a 1 is in column 2, the trace will be displayed on the CRT. This card is recommended for debugging programs.

APPENDIX C. AN/UYK-20 EMULATOR INSTRUCTION FORMAT

Format

Card Columns

5 6 7 8 9 10 11 12 13 14

RR, RL, opcode, 'f', 'a', 'm'

RI Type 2

5 6 7 8 9 10 11 12

```
RI Type 1 opcode , 'f' , 'd'
```

5 6 7 8 9 10 11 12 13 14 15 16 - 20

```
RK, RX      opcode , 'f' , ' a ' , ' m ' , ' y '
```

Notes:

- 1) All fields are in octal with the exceptions of addresses and the 'v' field which are decimal
- 2) All fields must be zero-filled
- 3) The following field restrictions must be followed:

Field	Range	Base
Opcode	00-76	octal
'f'	0-3	octal
'a'	00-17	octal
'm'	00-17	octal
's'	000-377	octal
'y'	00000-65535	decimal
addr	00000-65535	decimal

APPENDIX D. SAMPLE DEBUGGER OUTPUT

This appendix provides a sample program output illustrating the Trace option. The debugger is called at the beginning of every instruction fetch, and at the termination of the program. Space has been provided for dumping 48 memory addresses including emulated AN/UYK-20 registers and D-machine registers. Each output line consists of eight regions printed in hexadecimal with a total of eight hex digits (32 bits) per region.

The listing is annotated to indicate the identity of each output cell. A summary of cell definitions (numbering from left to right, top to bottom) follows:

DECIMAL ADDRESS	DESCRIPTION

0 - 15	General Register Stack 1
16 - 31	General Register Stack 2
32	Program Status word (PSW)
33	Breakpoint Register (BREAKPT)
34	Status Register 2 (SR2)
35	NEXTINSTR (next load address)
36	Clocktime
37	Real-Time Clock (RTC)
38	AMPCR
39	A1
40	A2
41	A3
42	MIR
43	BR1
44 - 47	Unused

END OF JOB

112

APPENDIX E. SAMPLE TEST PROGRAMS

The programs listed in this appendix were designed to illustrate how the AN/UYK-20 emulator can be used for developing programs. The first two programs presented are the generation of prime numbers and the solution of simultaneous linear equations by Cramer's Rule. They depict several AN/UYK-20 control structures such as looping, iteration, and condition code testing as well as numerous load, store, and arithmetic operations. The last program performs the Cramer's Rule algorithm and demonstrates input/output routines.

L	00034	PRIME NUMBER GENERATOR	
11		PRIME NUMBERS RETURNED IN GEN. REGISTERS	
P		THIS CARD WILL ENVOKE PAGING	
C	00206	FAULT INTERRUPT-EXECUTION ENDS *	
C	00214	NOT IMPLEMENTED-EXECUTION ENDS *	
C	00222	DIVIDE OVERFLOW - FAULT ENTERED *	
	63,0,C09,01	LITERAL LOAD, 1 INTO REG 4 (1ST PRIME)	00000005
	63,0,0,0,02	LITERAL LOAD, 2 INTO REG 5 (2ND PRIME)	00000420
	63,0,0,0,02	LITERAL LOAD, 2 INTO REG 6 (COUNTER)	00000000
	63,0,C07,05	LITERAL LOAD, NO. OF PRIMES INTO REG 7	00000000
	11,3,C09,0C,02018	STORE A 1 INTO ADDR 2048	00000000
	11,3,C05,00,02019	STORE A 2 INTO ADDR 2049	00000000
	24,0,C07,04	COMPARE N (NO. OF PRIMES) & 1	00000000
	44,1,C26	LOCAL JUMP, IF EQUAL	00000000
	24,0,0,07,05	COMPARE H AND 2	
	44,1,023	LOCAL JUMP, IF EQUAL	
	63,0,0,11,02	LITERAL LOAD, 2 INTO REG 11 (CANDIDATE)	
	62,2,11,01	LITERAL ADD, INCREMENT CANDIDATE	
	63,0,0,1,02	LITERAL LOAD, 2 INTO REG 10 (TRIAL DIVIS)	
	01,0,0,13,11	LOAD REG, CANDIDATE INTO REG 13	
	01,0,0,12,00	LOAD REG 12 WITH C	
	27,0,0,12,1C	DIVIDE REG, CANDIDATE BY TRIAL DIVISOR	
	24,0,C09,12	COMPARE, REMAINDER WITH 0	
	44,1,1371	JUMP IF EQUAL, NON-PRIME, GET NEXT CAND.	
	62,2,13,01	LITERAL ADD, INCREMENT TRIAL DIVISOR	
	24,0,0,10,11	COMPARE DIVISOR WITH CANDIDATE	
	47,1,1370	JUMP IF LESS THAN, TRY NEW DIVISOR	
	11,3,0,11,06,02018	STORE FOUND PRIME INTO MEMORY	
	62,2,0,5,01	LITERAL ADD, INCREMENT PRIME COUNTER	
	24,0,C05,07	COMPARE, PRIME COUNTER WITH N	
	47,1,1351	JUMP IF LESS THAN, GET NEXT PRIME CAND.	
	02,3,0,0,04,02019	LOAD MULT, PRIMES FROM MEMORY INTO REGS	
	63,0,0,17,00	PRIORITY INTERRUPT, HALT EXECUTION	
	03,0,0,01,02	LIT LOAD, FORCE FEED 2 INTO REG 01	
	63,0,0,0,01	LIT LOAD, FORCE FEED 1 INTO REG 00	
	03,0,0,17,00	PRIORITY INTERRUPT, HALT EXECUTION	
		GO, COMMENCE EXECUTION	
		MACHINE STATUS TO THE PRINTER	
S	00000001	00000005	00000007
H	00000002	00000005	00000007
	00000003	00000005	00000007
	00000004	00000005	00000007
	00000005	00000005	00000007
	00000006	00000005	00000007
	00000007	00000005	00000007
	00000008	00000005	00000007
	00000009	00000005	00000007
	00000010	00000005	00000007
	00000011	00000005	00000007
	00000012	00000005	00000007
	00000013	00000005	00000007
	00000014	00000005	00000007
	00000015	00000005	00000007
	00000016	00000005	00000007
	00000017	00000005	00000007
	00000018	00000005	00000007
	00000019	00000005	00000007
	00000020	00000005	00000007
	00000021	00000005	00000007
	00000022	00000005	00000007
	00000023	00000005	00000007
	00000024	00000005	00000007
	00000025	00000005	00000007
	00000026	00000005	00000007
	00000027	00000005	00000007
	00000028	00000005	00000007
	00000029	00000005	00000007
	00000030	00000005	00000007

END OF JOB

E

L	C0034	LINEAR EQUATIONS BY CRAMER'S RULE.		
I	I1	X IS RETURNED IN REG 01, Y IN REG 05		
C	C00206	FAULT INTERRUPT-EXECUTION ENDS		
C	C00214	NOT IMPLEMENTED-EXECUTION ENDS		
C	C0222	DIVIDE OVERFLOW - FAULT ENTERED		
D	D0248	BOTH EQUATIONS MUST BE IN STD FORM		
D	D0249	A (Y) PLUS B (X) EQUALS C		
D	D0250	Y COEFF OF 1ST EON		
D	D0251	X COEFF OF 1ST EON		
D	D0252	CONSTANT OF 1ST EON		
D	D0253	Y COEFF OF 2ND EON		
D	D0254	X COEFF OF 2ND EON		
D	D0255	CONSTANT OF 2ND EON		
C	C133	STORE X COEFF, EON 1, INTO REG 01		
C	C1303	STORE Y COEFF, EON 1, INTO REG 03		
C	C1305	STORE CONSTANT, EON 1, INTO REG 05		
C	C1311	STORE X COEFF, EON 2, INTO REG 11		
C	C1313	STORE Y COEFF, EON 2, INTO REG 13		
C	C1315	STORE CONSTANT, EON 2, INTO REG 15		
C	C260	MULT LEFT DIAG OF LENOM, RESULT IN 00/01		
C	C2601	MULT RT. DIAG OF LENOM, RESULT IN 10/11		
C	C210	SUBTRACT DOUBLE, RESULT OF DENOM IN 00/01		
C	C2100	COMPARE DENOM WITH ZERO		
C	C441	JUMP, IF EQUAL (ZERO DETERMINANT)		
C	C123	STORE DOUBLE, DENOM INTO 16/17		
C	C1230	LOAD 00 INTO REG 00		
C	C12301	RESTORE X COEFF, EON 1, INTO REG 01		
C	C12303	RESTORE Y COEFF, EON 1, INTO REG 03		
C	C12305	RESTORE CONSTANT OF EON 1 IN REG 05		
C	C12307	RESTORE X COEFF OF EON 2 IN REG 11		
C	C12309	RESTORE Y COEFF OF EON 2 IN REG 13		
C	C12311	RESTORE CONSTANT OF EON 2 IN REG 15		
C	C26010	X NUM LEFT DIAG MULT, RESULT IN 00/01		
C	C26011	X NUM RT. DIAG MULT, RESULT IN 10/11		
C	C26012	X NUM RESULT IN REG 00/01		
C	C26013	X RESIDES IN REG 01		
C	C26014	Y NUM LEFT DIAG MULT, RESULT REG 04/05		
C	C26015	Y NUM RT. DIAG MULT, RESULT REG 04/15		
C	C26016	DOUBLE SUBTRACT (FVAL NUM), RESULT 04/05		
C	C26017	DIVIDE NUM/DENOM, Y RESIDES IN 05		
C	C26018	PRIORITY INTERRUPT, HALT EXECUTION		
C	C26019	GO, COMMENCE EXECUTION		
C	C26020	MACHINE STATUS TO THE PRINTER		
C	C000000	C0000000	00000000	00000000
C	C000001	C0000001	00000001	00000001
C	C000002	C0000002	00000002	00000002
C	C000003	C0000003	00000003	00000003
C	C000004	C0000004	00000004	00000004
C	C000005	C0000005	00000005	00000005
C	C000006	C0000006	00000006	00000006
C	C000007	C0000007	00000007	00000007
C	C000008	C0000008	00000008	00000008
C	C000009	C0000009	00000009	00000009
C	C000010	C0000010	00000010	00000010
C	C000011	C0000011	00000011	00000011
C	C000012	C0000012	00000012	00000012
C	C000013	C0000013	00000013	00000013
C	C000014	C0000014	00000014	00000014
C	C000015	C0000015	00000015	00000015
C	C000016	C0000016	00000016	00000016
C	C000017	C0000017	00000017	00000017
C	C000018	C0000018	00000018	00000018
C	C000019	C0000019	00000019	00000019
C	C000020	C0000020	00000020	00000020
C	C000021	C0000021	00000021	00000021
C	C000022	C0000022	00000022	00000022
C	C000023	C0000023	00000023	00000023
C	C000024	C0000024	00000024	00000024
C	C000025	C0000025	00000025	00000025
C	C000026	C0000026	00000026	00000026
C	C000027	C0000027	00000027	00000027
C	C000028	C0000028	00000028	00000028
C	C000029	C0000029	00000029	00000029
C	C000030	C0000030	00000030	00000030
C	C000031	C0000031	00000031	00000031
C	C000032	C0000032	00000032	00000032
C	C000033	C0000033	00000033	00000033
C	C000034	C0000034	00000034	00000034
C	C000035	C0000035	00000035	00000035
C	C000036	C0000036	00000036	00000036
C	C000037	C0000037	00000037	00000037
C	C000038	C0000038	00000038	00000038
C	C000039	C0000039	00000039	00000039
C	C000040	C0000040	00000040	00000040
C	C000041	C0000041	00000041	00000041
C	C000042	C0000042	00000042	00000042
C	C000043	C0000043	00000043	00000043
C	C000044	C0000044	00000044	00000044
C	C000045	C0000045	00000045	00000045
C	C000046	C0000046	00000046	00000046
C	C000047	C0000047	00000047	00000047
C	C000048	C0000048	00000048	00000048
C	C000049	C0000049	00000049	00000049
C	C000050	C0000050	00000050	00000050
C	C000051	C0000051	00000051	00000051
C	C000052	C0000052	00000052	00000052
C	C000053	C0000053	00000053	00000053
C	C000054	C0000054	00000054	00000054
C	C000055	C0000055	00000055	00000055
C	C000056	C0000056	00000056	00000056
C	C000057	C0000057	00000057	00000057
C	C000058	C0000058	00000058	00000058
C	C000059	C0000059	00000059	00000059
C	C000060	C0000060	00000060	00000060
C	C000061	C0000061	00000061	00000061
C	C000062	C0000062	00000062	00000062
C	C000063	C0000063	00000063	00000063
C	C000064	C0000064	00000064	00000064
C	C000065	C0000065	00000065	00000065
C	C000066	C0000066	00000066	00000066
C	C000067	C0000067	00000067	00000067
C	C000068	C0000068	00000068	00000068
C	C000069	C0000069	00000069	00000069
C	C000070	C0000070	00000070	00000070
C	C000071	C0000071	00000071	00000071
C	C000072	C0000072	00000072	00000072
C	C000073	C0000073	00000073	00000073
C	C000074	C0000074	00000074	00000074
C	C000075	C0000075	00000075	00000075
C	C000076	C0000076	00000076	00000076
C	C000077	C0000077	00000077	00000077
C	C000078	C0000078	00000078	00000078
C	C000079	C0000079	00000079	00000079
C	C000080	C0000080	00000080	00000080
C	C000081	C0000081	00000081	00000081
C	C000082	C0000082	00000082	00000082
C	C000083	C0000083	00000083	00000083
C	C000084	C0000084	00000084	00000084
C	C000085	C0000085	00000085	00000085
C	C000086	C0000086	00000086	00000086
C	C000087	C0000087	00000087	00000087
C	C000088	C0000088	00000088	00000088
C	C000089	C0000089	00000089	00000089
C	C000090	C0000090	00000090	00000090
C	C000091	C0000091	00000091	00000091
C	C000092	C0000092	00000092	00000092
C	C000093	C0000093	00000093	00000093
C	C000094	C0000094	00000094	00000094
C	C000095	C0000095	00000095	00000095
C	C000096	C0000096	00000096	00000096
C	C000097	C0000097	00000097	00000097
C	C000098	C0000098	00000098	00000098
C	C000099	C0000099	00000099	00000099
C	C000100	C0000100	00000100	00000100
C	C000101	C0000101	00000101	00000101
C	C000102	C0000102	00000102	00000102
C	C000103	C0000103	00000103	00000103
C	C000104	C0000104	00000104	00000104
C	C000105	C0000105	00000105	00000105
C	C000106	C0000106	00000106	00000106
C	C000107	C0000107	00000107	00000107
C	C000108	C0000108	00000108	00000108
C	C000109	C0000109	00000109	00000109
C	C000110	C0000110	00000110	00000110
C	C000111	C0000111	00000111	00000111
C	C000112	C0000112	00000112	00000112
C	C000113	C0000113	00000113	00000113
C	C000114	C0000114	00000114	00000114
C	C000115	C0000115	00000115	00000115
C	C000116	C0000116	00000116	00000116
C	C000117	C0000117	00000117	00000117
C	C000118	C0000118	00000118	00000118
C	C000119	C0000119	00000119	00000119
C	C000120	C0000120	00000120	00000120
C	C000121	C0000121	00000121	00000121
C	C000122	C0000122	00000122	00000122
C	C000123	C0000123	00000123	00000123
C	C000124	C0000124	00000124	00000124
C	C000125	C0000125	00000125	00000125
C	C000126	C0000126	00000126	00000126
C	C000127	C0000127	00000127	00000127
C	C000128	C0000128	00000128	00000128
C	C000129	C0000129	00000129	00000129
C	C000130	C0000130	00000130	00000130
C	C000131	C0000131	00000131	00000131
C	C000132	C0000132	00000132	00000132
C	C000133	C0000133	00000133	00000133
C	C000134	C0000134	00000134	00000134
C	C000135	C0000135	00000135	00000135
C	C000136	C0000136	00000136	00000136
C	C000137	C0000137	00000137	00000137
C	C000138	C0000138	00000138	00000138
C	C000139	C0000139	00000139	00000139
C	C000140	C0000140	00000140	00000140
C	C000141	C0000141	00000141	00000141
C	C000142	C0000142	00000142	00000142
C	C000143	C0000143	00000143	00000143
C	C000144	C0000144	00000144	00000144
C	C000145	C0000145	00000145	00000145
C	C000146	C0000146	00000146	00000146
C	C000147	C0000147	00000147	00000147
C	C000148	C0000148	00000148	00000148
C	C000149	C0000149	00000149	00000149
C	C000150	C0000150	00000150	00000150
C	C000151	C0000151	00000151	00000151
C	C000152	C0000152	00000152	00000152
C	C000153	C0000153	00000153	00000153


```

P 00030
L 00230 FAULT INTERRUPT--EXECUTION ENDS *
C 00210 NOT IMPLEMENTED--EXECUTION ENDS *
C 00222 DIVIDE OVERFLOW - FAULT ENTERED!
I2 30 17650 10CM FOR 20 BUFFERS FROM CARD HEADER
I2 31 04076 10CM FOR 6 BUFFERS FROM CARD READER
I2 32 16758
I2 03 05120
I2 04 16705
I2 05 08076
I2 06 17655
I2 07 10240
I2 08 12336
I2 11 C0040
I2 15 00C30
D 02048 00C30
D 02049 C0030
D 02050 00C31
D 02C31 00C31
D 02032 00C33
D 02033 00031
D 02034 65545
D 02055 65545
63.0,0.15,0.1
03.0,0.13,0.5
12.1,0.0,1.7
35.0,0.7,0.0
12.1,0.2,1.7
35.0,0.3,0.0
01.2,0.12,0.0,0.1320
01.2,0.11,0.0,0.10240
15.1,0.10,1.1
02.0,0.12,1.1
45.1,1.375
01.2,0.14,0.0,0.4076
01.2,0.15,0.0,0.00C20
01.2,0.11,0.0,0.10240
05.1,0.15,1.4
15.1,0.15,1.1
02.0,0.13,1.1
45.1,1.374
22.2,0.11,0.0,0.00C26
01.2,0.13,0.0,0.00C40
02.0,0.13,1.1
45.1,1.356
12.1,0.15,1.7
35.0,0.3,0.0

```

BOTH EQUATIONS MUST BE IN STD FORM
A (Y) PLUS B (X) EQUALS C
Y COEFF OF 1ST EON
X COEFF OF 1ST EON
CONSTANT OF 1ST EON
Y COEFF OF 2ND EON
X COEFF OF 2ND EON
CONSTANT OF 2ND EON
LOAD 1 INTO REG. 12
LOAD S91 WITH REG. 12
STORE DOUBLE INTO 10CM
INPUT 14 BUFFERS FROM CARD READER
STORE DOUBLE INTO 10CM
INPUT 6 BUFFERS FROM THE CARD READER
INITIALIZE BUFFER WORDS COUNT (20 PRT BUFFERS)
INITIALIZE OUTPUT BUFFER ADDR
STORE (RA) INTO Y* AND INDEX BY 1 (STORE SPACES)
DECREMENT BUFFER COUNTER
COUNTER EQUAL ZERO
INITIALIZE BUFFER ADDRESS OF INPUT
BUFFER COUNT
INITIALIZE DESTINATION BUFFER ADDR
LOAD AND INDEX BY 1
STORE RA INTO Y* AND INDEX Y*
DECREMENT COUNTER
COUNTER EQUAL ZERO
ADD 26 TO BUFFER ADDR
RESET - BUFFER WORD COUNT
DECREMENT NEXT COUNTER (NO. OF BUFFERS)
COUNTER EQUAL ZERO
STORE DOUBLE INTO 10CM
OUTPUT 14 BUFFERS TO PRT


```

63+0+13+01
03+0+13+05
01+2+11+00+00076
01+2+12+00+00030
01+2+13+00+00035
01+2+14+00+00120
01+2+15+00+00120
15+1+10+11
02+0+12+11
45+1+375
01+2+11+00+00076
05+1+15+14
15+1+15+11
02+0+13+11
45+1+374
22+2+11+00+00026
01+2+13+00+00040
02+0+13+11
45+1+356
63+0+13+00
03+0+13+05
01+3+01+00+00030
01+3+03+00+00031
01+3+05+00+00032
01+3+11+00+00033
01+3+13+00+00034
22+2+10+00+00035
26+0+13+03
21+0+00+10
25+3+00+17+00240
44+1+052
12+3+07+06+00014
01+2+02+00+00030
01+3+01+00+00030
01+3+03+00+00031
01+3+05+00+00032
01+3+11+00+00033
01+3+13+00+00034
26+0+00+15
26+0+13+05
21+0+00+10
27+0+00+17
26+0+00+13
26+0+10+03
21+0+03+14
27+0+03+17

***RE-ENTRY POINT FOR REPETITIVE EXECUTION***
LOAD SRI WITH A 1 FROM REG 13
INITIALIZE NEXT OUTPUT BUFFER ADDRESS
INITIALIZE NEXT BUFFER WORD COUNT (5 PRT BUFFERS)
BUFFER COUNT
INITIALIZE BUFFER ADDR OF SECOND INPUT BUFFER
STORE (RA) INTO Y* AND INDEX BY 1 (STORE SPACES)
DECREMENT GUFER COUNTER
COUNTER EQUAL ZERO
REINITIALIZE BUFFER ADDR IN REG 9
LOAD AND INDEX BY J
STORE RA INTO Y* AND INDEX Y*
DECREMENT COUNTER
COUNTER EQUAL ZERO
ADD 26 TO BUFFER ADDR
RESET BUFFER WORD COUNT
DECREMENT NEXT COUNTER (NO. OF BUFFERS)
COUNTER EQUAL ZERO
LOAD REG 13 WITH 0
LOAD SRI WITH 0
***CRAMER'S RULE ALGORITHM*** STORE X OF EON 1
STORE Y COEFF, EON 1, INTO REG 03
STORE CONSTANT, EON 1, INTO REG 05
STORE X COEFF, EON 2, INTO REG 11
STORE X COEFF, EON 2, INTO REG 13
STORE CONSTANT, EON 2, INTO REG 15
NUM LEFT DIAG OF DENOM, RESULT IN 00/01
MULT RT, DIAG OF DENOM, RESULT IN 10/11
SUBTRACT DOUBLE, RESULT OF DENOM IN 00/01
COMPARE DENOM WITH ZERO
JUMP, IF EQUAL (ZERO DETERMINANT)
STORE DOUBLE, DENOM INTO 16/17
LOAD 00 INTO REG 00
RESTORE Y COEFF, EON 1, INTO REG 01
RESTORE X COEFF, EON 1, INTO REG 03
RESTORE CONSTANT OF EON 1 IN REG 05
RESTORE Y COEFF OF EON 2 IN REG 11
RESTORE X COEFF OF EON 2 IN REG 13
RESTORE CONSTANT OF EON 2 IN REG 15
X NUM LEFT DIAG MULT, RESULT IN 00/01
X NUM RT, 01AG MULT, RESULT IN 10/11
X NUM RESULT IN REG 00/01
X RESIDES IN REG 01
Y NUM LEFT DIAG MULT, RESULT REG 04/05
Y NUM RT, DIAG MULT, RESULT REG 14/15
DOUBLE SUBTRACT (EVAL NUM), RESULT 04/05
DIVIDE NUM/DENOM, Y RESIDES IN 05

```



```

01:2:07:00:16359
63:0:03:00
63:3:09:12
1C:3:03:07:00030
44:1:073
63:0:07:00
62:0:07:01
40:1:371
01:2:07:00:16633
63:0:03:00
63:3:03:12
1C:3:03:07:00070
44:1:023
63:0:03:00
62:0:07:01
40:1:371
63:0:15:01
03:0:15:05
12:1:04:17
35:0:03:00
03:0:05:00
63:0:15:01
03:0:15:05
2C:2:11:00:00178
01:2:13:00:00040
05:1:15:14
15:1:15:11
02:0:13:11
45:1:374
12:1:04:17
35:0:03:00
03:0:05:00

***INT TO BCL CONVERSION*** TWICE BUFFER ADDR PLUS 1
LIT LOAD, C INTO REG 0
LIT DIVIDE, REG 0/1 BY 10
BYTE STORE, REG 7 MUST CONTAIN TWICE ADDR
LOCAL JUMP EQUAL (ZERO QUOTIENT)
LIT LOAD, C INTO REG 0
LIT SUBTRACT, 1 FROM REG 7
LOCAL JUMP
LOAD CONSTANT, TWICE ADDR PLUS 1 IN REG. 7
LIT LOAD, C INTO REG 4
LIT DIVIDE, REG 4/5 BY 10
BYTE JUMP
JUMP EQUAL, CHECK FOR ZERO QUOTIENT
LIT LOAD, C INTO REG 4
LIT SUBTRACT, 1 FROM REG 7
LOCAL JUMP
LOAD REG 13 WITH A 1
LOAD SR1 WITH A 1
STORE DOUBLE INTO 10CM
DUMP 5 BUFFERS TO PRI
HALT

***NO SOLUTION CASE***
LOAD SR1 WITH A 1
RECOMPUTE OUTPUT BUFFER ADDR FOR *NO SOLUTION*
REINITIALIZE COUNTER
LOAD AND INDEX BY 1
STORE RA INTO Y* AND INDEX Y*
DECREMENT COUNTER
COUNTER EQUAL ZERO
STORE DOUBLE INTO 10CM
DUMP 5 BUFFERS TO PRI
HALT

```


THIS IS A DEMONSTRATION PROGRAM OF THE AN/JYK20 EMULATOR.
THE PROGRAM SOLVES TWO SIMULTANEOUS EQUATIONS BY CRAMER'S RULE.

THE INPUT COEFFICIENT VALUES AND CONSTANTS FOR EACH EQUATION MUST BE INSERTED
USING LOADER CONTROL CARDS IN THE FOLLOWING LOCATIONS

1ST EQUATION. A1 IN LOCATION 2050 D.
B1 IN LOCATION 2051 D.
CONSTANT IV LOCATION 2052 D.

2ND EQUATION. A2 IN LOCATION 2053 D.
B2 IN LOCATION 2054 D.
CONSTANT IV LOCATION 2055 D.

THE RESULT IS.
X > 2

Y > 1
TO CHANGE VARIABLES INSERT LOADER JCL HERE, OTHERWISE EXIT
11 00C30 REINITIALIZE BUFFER WORD COUNT FOR BUFFER INIT.
12 02053 Y COEFF OF 2ND EQUATION
13 00C31 X COEFF OF 2ND EQUATION
14 02054 Y COEFF OF 1ST EQUATION
15 02055 CONSTANT OF 2ND EQUATION
16 00C31 RE-EXECUTE PROGRAM AT ADDR 1055 DECIMAL

THE RESULT IS.
X > 1
NO SOLUTION

Y > 1
TO CHANGE VARIABLES INSERT LOADER JCL HERE, OTHERWISE EXIT
E END EXECUTION

APPENDIX F. EMULATOR LISTING

This appendix provides a listing of the TRANSLANG assembler output of the AN/UYK-20 emulation. A source file copy of the emulator exists on disk as well as on cards. A microinstruction object file is also maintained on disk.

The listing is divided into four sections. The left most section contains the microinstruction address composed of four hexadecimal digits followed by a 56-bit microinstruction created from a TRANSLANG instruction. The center section contains the TRANSLANG source which includes labels, an instruction, and/or a comment field. The final number printed on the right most side of the listing is the sequence number of the TRANSLANG source statement. This number, printed in decimal, is created by a Burroughs software utility program called CARD-LIST. This number must be used when editing source programs on disk.


```

%098 4809 0C43 0C3C 00FC
%099 4809 0C0C 003C 00FC
%09A 4809 00C0 0031 00FC
%09B 00C0 00C3 0070 00C0

%09C 4809 4800 9C3C 00FC
%09D 4809 00C0 0C30 00C0
%09E 4809 48C1 4C70 00FC
%09F 48C9 4C41 1830 00FC
%0A0 70C0 0000 0030 00C0
%0A1 4809 4C41 0832 00FC
%0A2 3000 00C0 0C30 00C0
%0A3 4809 4C43 0C70 00FC
%0A4 3C19 4800 9C3C 00FC
%0A5 0980 00C3 0030 0040
%0A6 300C 00C0 0030 003C
%0A7 4809 4C40 0830 00FC
%0A8 4820 00C3 0030 00FC

%0A9 051C 0000 0030 00C0
%0AA 4809 0C40 1030 00FC
%0AB 4809 20C3 001C 00FC
%0AC 0700 00C3 0030 00C0
%0AD 046C 00C3 0C30 00C0
%0AE 4809 0C43 4C30 00FC
%0AF 0870 00C3 00C0 00C0
%0B0 4809 48C0 2030 00FC
%0B1 097C 0000 0000 00C0
%0B2 4809 0C40 0C30 00FC
%0B3 4809 0C12 003C 00FC
%0B4 5808 00C0 003C 00FC
%0B5 010C 0000 0070 0040
%0B6 4809 0C03 001C 00FC
%0B7 05EC 0003 0070 0040

%0B8 4809 2003 201C 00C0
%0B9 0230 00C3 0030 00C0
%0BA 0460 00C0 0030 00C0
%0BB 4809 0C43 001C 00FC
%0BC 0810 00C3 00C0 00C0
%0BD 4809 0C45 0030 00FC
%0BE 4809 0C03 001C 00FC
%0BF 05E0 00C3 0030 0040

%0C0 4809 00C0 9C3C 00FC
%0C1 4809 00C0 0C30 00C0
%0C2 4809 00C0 0031 00FC
%0C3 00C0 00C3 0070 00C0

%0C4 4809 4800 9C3C 00FC
%0C5 4809 00C0 0C30 00C0
%0C6 4809 48C1 4C70 00FC
%0C7 48C9 4C41 1830 00FC
%0C8 70C0 0000 0030 00C0
%0C9 4809 4C41 0832 00FC
%0CA 3000 00C0 0C30 00C0
%0CB 4809 4C43 0C70 00FC
%0CC 3C19 4800 9C3C 00FC
%0CD 0980 00C3 0030 0040
%0CE 300C 00C0 0030 003C
%0CF 4809 4C40 0830 00FC
%0D0 4820 00C3 0030 00FC

%0D1 051C 0000 0030 00C0
%0D2 4809 0C40 1030 00FC
%0D3 4809 20C3 001C 00FC
%0D4 0700 00C3 0030 00C0
%0D5 046C 00C3 0C30 00C0
%0D6 4809 0C43 4C30 00FC
%0D7 0870 00C3 00C0 00C0
%0D8 4809 48C0 2030 00FC
%0D9 097C 0000 0000 00C0
%0DA 4809 0C40 0C30 00FC
%0DB 4809 0C12 003C 00FC
%0DC 5808 00C0 003C 00FC
%0DD 010C 0000 0070 0040
%0DE 4809 0C03 001C 00FC
%0DF 05EC 0003 0070 0040

%0E0 4809 2003 201C 00C0
%0E1 0230 00C3 0030 00C0
%0E2 0460 00C0 0030 00C0
%0E3 4809 0C43 001C 00FC
%0E4 0810 00C3 00C0 00C0
%0E5 4809 0C45 0030 00FC
%0E6 4809 0C03 001C 00FC
%0E7 05E0 00C3 0030 0040

%0E8 4809 00C0 9C3C 00FC
%0E9 4809 00C0 0C30 00C0
%0EA 4809 00C0 0031 00FC
%0EB 00C0 00C3 0070 00C0

%0EC 4809 4800 9C3C 00FC
%0ED 4809 00C0 0C30 00C0
%0EE 4809 48C1 4C70 00FC
%0EF 48C9 4C41 1830 00FC
%0F0 70C0 0000 0030 00C0
%0F1 4809 4C41 0832 00FC
%0F2 3000 00C0 0C30 00C0
%0F3 4809 4C43 0C70 00FC
%0F4 3C19 4800 9C3C 00FC
%0F5 0980 00C3 0030 0040
%0F6 300C 00C0 0030 003C
%0F7 4809 4C40 0830 00FC
%0F8 4820 00C3 0030 00FC

```

```

%0F9 051C 0000 0030 00C0
%0FA 4809 0C40 1030 00FC
%0FB 4809 20C3 001C 00FC
%0FC 0700 00C3 0030 00C0
%0FD 046C 00C3 0C30 00C0
%0FE 4809 0C43 4C30 00FC
%0FF 0870 00C3 00C0 00C0
%100 4809 48C0 2030 00FC
%101 097C 0000 0000 00C0
%102 4809 0C40 0C30 00FC
%103 4809 0C12 003C 00FC
%104 5808 00C0 003C 00FC
%105 010C 0000 0070 0040
%106 4809 0C03 001C 00FC
%107 05EC 0003 0070 0040

%108 4809 2003 201C 00C0
%109 0230 00C3 0030 00C0
%10A 0460 00C0 0030 00C0
%10B 4809 0C43 001C 00FC
%10C 0810 00C3 00C0 00C0
%10D 4809 0C45 0030 00FC
%10E 4809 0C03 001C 00FC
%10F 05E0 00C3 0030 0040

%110 4809 00C0 9C3C 00FC
%111 4809 00C0 0C30 00C0
%112 4809 00C0 0031 00FC
%113 00C0 00C3 0070 00C0

%114 4809 4800 9C3C 00FC
%115 4809 00C0 0C30 00C0
%116 4809 48C1 4C70 00FC
%117 48C9 4C41 1830 00FC
%118 70C0 0000 0030 00C0
%119 4809 4C41 0832 00FC
%11A 3000 00C0 0C30 00C0
%11B 4809 4C43 0C70 00FC
%11C 3C19 4800 9C3C 00FC
%11D 0980 00C3 0030 0040
%11E 300C 00C0 0030 003C
%11F 4809 4C40 0830 00FC
%120 4820 00C3 0030 00FC

```

```

%121 051C 0000 0030 00C0
%122 4809 0C40 1030 00FC
%123 4809 20C3 001C 00FC
%124 0700 00C3 0030 00C0
%125 046C 00C3 0C30 00C0
%126 4809 0C43 4C30 00FC
%127 0870 00C3 00C0 00C0
%128 4809 48C0 2030 00FC
%129 097C 0000 0000 00C0
%12A 4809 0C40 0C30 00FC
%12B 4809 0C12 003C 00FC
%12C 5808 00C0 003C 00FC
%12D 010C 0000 0070 0040
%12E 4809 0C03 001C 00FC
%12F 05EC 0003 0070 0040

%130 4809 2003 201C 00C0
%131 0230 00C3 0030 00C0
%132 0460 00C0 0030 00C0
%133 4809 0C43 001C 00FC
%134 0810 00C3 00C0 00C0
%135 4809 0C45 0030 00FC
%136 4809 0C03 001C 00FC
%137 05E0 00C3 0030 0040

%138 4809 00C0 9C3C 00FC
%139 4809 00C0 0C30 00C0
%13A 4809 00C0 0031 00FC
%13B 00C0 00C3 0070 00C0

%13C 4809 4800 9C3C 00FC
%13D 4809 00C0 0C30 00C0
%13E 4809 48C1 4C70 00FC
%13F 48C9 4C41 1830 00FC
%140 70C0 0000 0030 00C0
%141 4809 4C41 0832 00FC
%142 3000 00C0 0C30 00C0
%143 4809 4C43 0C70 00FC
%144 3C19 4800 9C3C 00FC
%145 0980 00C3 0030 0040
%146 300C 00C0 0030 003C
%147 4809 4C40 0830 00FC
%148 4820 00C3 0030 00FC

```

```

%149 051C 0000 0030 00C0
%14A 4809 0C40 1030 00FC
%14B 4809 20C3 001C 00FC
%14C 0700 00C3 0030 00C0
%14D 046C 00C3 0C30 00C0
%14E 4809 0C43 4C30 00FC
%14F 0870 00C3 00C0 00C0
%150 4809 48C0 2030 00FC
%151 097C 0000 0000 00C0
%152 4809 0C40 0C30 00FC
%153 4809 0C12 003C 00FC
%154 5808 00C0 003C 00FC
%155 010C 0000 0070 0040
%156 4809 0C03 001C 00FC
%157 05EC 0003 0070 0040

%158 4809 2003 201C 00C0
%159 0230 00C3 0030 00C0
%15A 0460 00C0 0030 00C0
%15B 4809 0C43 001C 00FC
%15C 0810 00C3 00C0 00C0
%15D 4809 0C45 0030 00FC
%15E 4809 0C03 001C 00FC
%15F 05E0 00C3 0030 0040

%160 4809 00C0 9C3C 00FC
%161 4809 00C0 0C30 00C0
%162 4809 00C0 0031 00FC
%163 00C0 00C3 0070 00C0

%164 4809 4800 9C3C 00FC
%165 4809 00C0 0C30 00C0
%166 4809 48C1 4C70 00FC
%167 48C9 4C41 1830 00FC
%168 70C0 0000 0030 00C0
%169 4809 4C41 0832 00FC
%16A 3000 00C0 0C30 00C0
%16B 4809 4C43 0C70 00FC
%16C 3C19 4800 9C3C 00FC
%16D 0980 00C3 0030 0040
%16E 300C 00C0 0030 003C
%16F 4809 4C40 0830 00FC
%170 4820 00C3 0030 00FC

```


OPCODE	OP	OPN	OPD	OPC	OPR	OPR2	OPR3	OPR4	OPR5	OPR6	OPR7	OPR8	OPR9	OPR10	OPR11	OPR12	OPR13	OPR14	OPR15	OPR16	OPR17	OPR18	OPR19	OPR20	OPR21	OPR22	OPR23	OPR24	OPR25	OPR26	OPR27	OPR28	OPR29	OPR30	OPR31	OPR32	OPR33	OPR34	OPR35	OPR36	OPR37	OPR38	OPR39	OPR40	OPR41	OPR42	OPR43	OPR44	OPR45	OPR46	OPR47	OPR48	OPR49	OPR50	OPR51	OPR52	OPR53	OPR54	OPR55	OPR56	OPR57	OPR58	OPR59	OPR60	OPR61	OPR62	OPR63	OPR64	OPR65	OPR66	OPR67	OPR68	OPR69	OPR70	OPR71	OPR72	OPR73	OPR74	OPR75	OPR76	OPR77	OPR78	OPR79	OPR80	OPR81	OPR82	OPR83	OPR84	OPR85	OPR86	OPR87	OPR88	OPR89	OPR90	OPR91	OPR92	OPR93	OPR94	OPR95	OPR96	OPR97	OPR98	OPR99	OPR100	OPR101	OPR102	OPR103	OPR104	OPR105	OPR106	OPR107	OPR108	OPR109	OPR110	OPR111	OPR112	OPR113	OPR114	OPR115	OPR116	OPR117	OPR118	OPR119	OPR120	OPR121	OPR122	OPR123	OPR124	OPR125	OPR126	OPR127	OPR128	OPR129	OPR130	OPR131	OPR132	OPR133	OPR134	OPR135	OPR136	OPR137	OPR138	OPR139	OPR140	OPR141	OPR142	OPR143	OPR144	OPR145	OPR146	OPR147	OPR148	OPR149	OPR150	OPR151	OPR152	OPR153	OPR154	OPR155	OPR156	OPR157	OPR158	OPR159	OPR160	OPR161	OPR162	OPR163	OPR164	OPR165	OPR166	OPR167	OPR168	OPR169	OPR170	OPR171	OPR172	OPR173	OPR174	OPR175	OPR176	OPR177	OPR178	OPR179	OPR180	OPR181	OPR182	OPR183	OPR184	OPR185	OPR186	OPR187	OPR188	OPR189	OPR190	OPR191	OPR192	OPR193	OPR194	OPR195	OPR196	OPR197	OPR198	OPR199	OPR200	OPR201	OPR202	OPR203	OPR204	OPR205	OPR206	OPR207	OPR208	OPR209	OPR210	OPR211	OPR212	OPR213	OPR214	OPR215	OPR216	OPR217	OPR218	OPR219	OPR220	OPR221	OPR222	OPR223	OPR224	OPR225	OPR226	OPR227	OPR228	OPR229	OPR230	OPR231	OPR232	OPR233	OPR234	OPR235	OPR236	OPR237	OPR238	OPR239	OPR240	OPR241	OPR242	OPR243	OPR244	OPR245	OPR246	OPR247	OPR248	OPR249	OPR250	OPR251	OPR252	OPR253	OPR254	OPR255	OPR256	OPR257	OPR258	OPR259	OPR260	OPR261	OPR262	OPR263	OPR264	OPR265	OPR266	OPR267	OPR268	OPR269	OPR270	OPR271	OPR272	OPR273	OPR274	OPR275	OPR276	OPR277	OPR278	OPR279	OPR280	OPR281	OPR282	OPR283	OPR284	OPR285	OPR286	OPR287	OPR288	OPR289	OPR290	OPR291	OPR292	OPR293	OPR294	OPR295	OPR296	OPR297	OPR298	OPR299	OPR300	OPR301	OPR302	OPR303	OPR304	OPR305	OPR306	OPR307	OPR308	OPR309	OPR310	OPR311	OPR312	OPR313	OPR314	OPR315	OPR316	OPR317	OPR318	OPR319	OPR320	OPR321	OPR322	OPR323	OPR324	OPR325	OPR326	OPR327	OPR328	OPR329	OPR330	OPR331	OPR332	OPR333	OPR334	OPR335	OPR336	OPR337	OPR338	OPR339	OPR340	OPR341	OPR342	OPR343	OPR344	OPR345	OPR346	OPR347	OPR348	OPR349	OPR350	OPR351	OPR352	OPR353	OPR354	OPR355	OPR356	OPR357	OPR358	OPR359	OPR360	OPR361	OPR362	OPR363	OPR364	OPR365	OPR366	OPR367	OPR368	OPR369	OPR370	OPR371	OPR372	OPR373	OPR374	OPR375	OPR376	OPR377	OPR378	OPR379	OPR380	OPR381	OPR382	OPR383	OPR384	OPR385	OPR386	OPR387	OPR388	OPR389	OPR390	OPR391	OPR392	OPR393	OPR394	OPR395	OPR396	OPR397	OPR398	OPR399	OPR400	OPR401	OPR402	OPR403	OPR404	OPR405	OPR406	OPR407	OPR408	OPR409	OPR410	OPR411	OPR412	OPR413	OPR414	OPR415	OPR416
--------	----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------


```

0174 4809 EC5C 0B0C 00FC
0175 4809 00F1 2C00 00F0
0176 4809 00F0 2C00 00F0
0177 4809 00F0 2C00 00F0
0178 1440 07F3 0000 C0C0
0179 05E7 00C0 0C00 C040

017A 4809 20C3 001C C0F0
017B 420C 0000 0C00 00A0
017C 046C 00C3 0000 0060
017D 4809 0C41 8B7C 40F0
017E 4809 2C5F 8800 00FC
017F 0010 0000 0C00 C0E0
0180 4809 0C40 00A0 C0F0
0181 05E0 0C00 0C30 0040

0182 2809 0000 0030 00F0
0183 3809 00C0 0C70 00F0
0184 0510 00C0 0C70 00F0
0185 8A09 20C3 001C C0F0
0186 0230 0000 0C30 00F0
0187 3809 0C32 00C0 00F0
0188 3A0E 00C0 0C7C 00F0
0189 0460 0C00 0C7C 00F0
018A 4809 0C43 4C30 C0F0
018B 4809 20C3 001C C0F0
018C 07C0 0C00 0070 00FC
018D 046C 0000 0C30 00F0
018E 0030 00C3 0030 00EC
018F 4812 0C43 0071 00F0
0190 4809 0C41 8032 00F0
0191 0000 0C00 0000 0010
0192 4809 2C5E 273C 00F0
0193 03C0 00C3 003C 00F0
0194 4809 C132 00C0 00F0
0195 00F0 0000 0C00 00F0
0196 48C9 6C73 0C30 00F0
0197 0210 00C0 0030 00EC
0198 00C0 00C3 003C 00F0
0199 8023 00C3 003C 00F0
019A 80C9 00F0 1007 00F0
019B 80C9 ACB1 001C 00F0
019C 001C 00C0 0C30 00F0
019D 4809 ACCB 4C30 00FC
019E 280B 0C00 0C00 00F0
019F 027F 0C00 0000 004C
01A0 4809 0C03 001C 00F0
01A1 4809 E159 0C30 00F0
01A2 0800 00C3 003C 00FC
01A3 700B 0C00 0000 00FC

SETPAGING:
LIT = MAR2
PSW = LIT, 22 = SAR
INPUT - 1 = CPCCR
B C = B, CSAR
LIT OR B C = B
1 = LIT
D = MIR
OUT - 1 = MPCCR

CHARSTRING:
LIT = MAR2
THIS ROUTINE WILL READ THE CHARACTER
STRING STARTING IN COL. 9 UNTIL FINDING COL. 9
A QUOTE (*). THE STRING WILL THEN BE
STORED AT THE ADDRESS IN COL. 4-8. IF
COL. 4-8 IS FLANK, THE STORAGE ADDRESS
WILL DEFAULT TO THE ADDRESS IN NEXTINSTR.
A CLEAR CONDITION ELTS
GET ADDRESS IN COL. 4-8
MAR2 CONTAINS ADDR OF NEXT INSTRUCTION
CHECK FOR BLANK ADDRESS FIELD
STEP ELSE SKIP
READ CONTENTS OF NEXTINSTR
AL CONTAINS ADDR TO BE WRITTEN INTO
SET UP READ ADDR OF COL. 9-12 IN MAR2
READ A 4 CHARACTER BLOCK
SET UP COUNTER FOR 4 ITERATIONS
SHIFT 1 CHARACTER INTO B TO BE ANALYZED
EXTRACT RIGHTMOST CHARACTER
SET UP 6 BIT MASK
TEST FOR = (CF)
IF A (*) IS IN THE STRING SET LCI
A ADD VALUE TO (*) SO THAT SPACE APPEARS
B, MIR, SET LCI
CHECK IF LOOP DONE 4 TIMES
CHECK NEXT ADDRESS
MIR2 NO WAS VALUE ADDRESS
WRITE 4 CHAR BLOCK TO MEMORY
INCREMENT WRITE ADDRESS
CHECK IF (*) WAS FOUND
ENDCHARSTRING - 1 = MPCCR
A3 AND MAR2 HAVE READ ADDRESS
A3 = MAR2
CHECK IF YOU ARE AT THE END OF THE
CARD077XBC = LIT
IF TRUE THEN SKIP ELSE SKIP

```



```

0000 0000 0000 0030 00F0 28 = SAV
0001 4000 0000 0040 00F0 A1 OR 1 = A1,CSAF
0002 4000 0000 0040 00F0 A1 C = A1
0003 4009 A0C1 0030 00F0 JUMP
0004 4820 0003 0030 00F0

$ ***** END CARRY ROUTINES *****
CLEARBUF:
0005 0640 2C30 00F0 AMPCR = A2
0006 0000 0931 00F0 0 = BC0,ALCTR
0007 01F0 0003 0040 00F0 COMP 16 = SARI31 = LIT % REQUIRED FOR BCB
0008 20C3 001C 00F0 LIT = MAR2
0009 0003 0030 00F0 PRINTBUFF = LIT
0010 0003 0030 00F0 B = HIR
0011 0003 0030 00F0 EDUPUT - 1 = CPCR
0012 0003 0030 00F0 IF NOT C0V THEN PHAR + 1 = MAR2,INC1 STEF ELSE SKIP
0013 0003 0030 00F0 PBUFF - 1 = HPCR
0014 0003 0030 00F0 A2 = AMPCR
0015 0003 0030 00F0 STEP
0016 0003 0030 00F0 JUMP
0017 0003 0030 00F0

$ ***** CONDITION CODE SETTING *****
SETCC:
0018 0003 0030 00F0 A2 EOL B
0019 0003 0030 00F0 IF FALSE THEN A2 XOR B1 SKIP
0020 0003 0030 00F0 SET00 - 1 = MPCR
0021 0003 0030 00F0 IF HST THEN SKIP
0022 0003 0030 00F0 LIKE - 1 = MPCR
0023 0003 0030 00F0 A2
0024 0003 0030 00F0 IF HST THEN STEP ELSE SKIP % HST -> A2 WAS NEGATIVE
0025 0003 0030 00F0 SET11 - 1 = MPCR
0026 0003 0030 00F0 SET01 - 1 = MPCR
0027 0003 0030 00F0 A2 GTR B
0028 0003 0030 00F0 LIKE
0029 0003 0030 00F0 IF TRUE THEN STEP ELSE SKIP
0030 0003 0030 00F0 SET01 - 1 = MPCR
0031 0003 0030 00F0 SET11 - 1 = MPCR
0032 0003 0030 00F0 SET11 - 1 = MPCR
0033 0003 0030 00F0

2819 0003 0030 00F0 IF LCL THEN SKIP
0034 0003 0030 00F0 B L = 0
0035 0003 0030 00F0 COMP 16 = SAC
0036 0003 0030 00F0 IF FALSE THEN B1 SKIP
0037 0003 0030 00F0 SET00 - 1 = MPCR
0038 0003 0030 00F0 IF HST THEN SKIP
0039 0003 0030 00F0 SET01 - 1 = MPCR
0040 0003 0030 00F0 SET11 - 1 = MPCR
0041 0003 0030 00F0 SET11 - 1 = MPCR
0042 0003 0030 00F0

0043 0003 0030 00F0
0044 0003 0030 00F0
0045 0003 0030 00F0
0046 0003 0030 00F0
0047 0003 0030 00F0
0048 0003 0030 00F0
0049 0003 0030 00F0
0050 0003 0030 00F0
0051 0003 0030 00F0
0052 0003 0030 00F0
0053 0003 0030 00F0
0054 0003 0030 00F0
0055 0003 0030 00F0
0056 0003 0030 00F0
0057 0003 0030 00F0
0058 0003 0030 00F0
0059 0003 0030 00F0
0060 0003 0030 00F0
0061 0003 0030 00F0
0062 0003 0030 00F0
0063 0003 0030 00F0
0064 0003 0030 00F0
0065 0003 0030 00F0
0066 0003 0030 00F0
0067 0003 0030 00F0
0068 0003 0030 00F0
0069 0003 0030 00F0
0070 0003 0030 00F0
0071 0003 0030 00F0
0072 0003 0030 00F0
0073 0003 0030 00F0
0074 0003 0030 00F0
0075 0003 0030 00F0
0076 0003 0030 00F0
0077 0003 0030 00F0
0078 0003 0030 00F0
0079 0003 0030 00F0
0080 0003 0030 00F0
0081 0003 0030 00F0
0082 0003 0030 00F0
0083 0003 0030 00F0
0084 0003 0030 00F0
0085 0003 0030 00F0
0086 0003 0030 00F0
0087 0003 0030 00F0
0088 0003 0030 00F0
0089 0003 0030 00F0
0090 0003 0030 00F0
0091 0003 0030 00F0
0092 0003 0030 00F0
0093 0003 0030 00F0
0094 0003 0030 00F0
0095 0003 0030 00F0
0096 0003 0030 00F0
0097 0003 0030 00F0
0098 0003 0030 00F0
0099 0003 0030 00F0
0100 0003 0030 00F0
0101 0003 0030 00F0
0102 0003 0030 00F0
0103 0003 0030 00F0
0104 0003 0030 00F0
0105 0003 0030 00F0
0106 0003 0030 00F0
0107 0003 0030 00F0
0108 0003 0030 00F0
0109 0003 0030 00F0
0110 0003 0030 00F0
0111 0003 0030 00F0
0112 0003 0030 00F0
0113 0003 0030 00F0
0114 0003 0030 00F0
0115 0003 0030 00F0
0116 0003 0030 00F0
0117 0003 0030 00F0
0118 0003 0030 00F0
0119 0003 0030 00F0
0120 0003 0030 00F0
0121 0003 0030 00F0
0122 0003 0030 00F0
0123 0003 0030 00F0
0124 0003 0030 00F0
0125 0003 0030 00F0
0126 0003 0030 00F0
0127 0003 0030 00F0
0128 0003 0030 00F0
0129 0003 0030 00F0
0130 0003 0030 00F0
0131 0003 0030 00F0
0132 0003 0030 00F0
0133 0003 0030 00F0
0134 0003 0030 00F0
0135 0003 0030 00F0
0136 0003 0030 00F0
0137 0003 0030 00F0
0138 0003 0030 00F0
0139 0003 0030 00F0
0140 0003 0030 00F0
0141 0003 0030 00F0
0142 0003 0030 00F0
0143 0003 0030 00F0
0144 0003 0030 00F0
0145 0003 0030 00F0
0146 0003 0030 00F0
0147 0003 0030 00F0
0148 0003 0030 00F0
0149 0003 0030 00F0
0150 0003 0030 00F0
0151 0003 0030 00F0
0152 0003 0030 00F0
0153 0003 0030 00F0
0154 0003 0030 00F0
0155 0003 0030 00F0
0156 0003 0030 00F0
0157 0003 0030 00F0
0158 0003 0030 00F0
0159 0003 0030 00F0
0160 0003 0030 00F0
0161 0003 0030 00F0
0162 0003 0030 00F0
0163 0003 0030 00F0
0164 0003 0030 00F0
0165 0003 0030 00F0
0166 0003 0030 00F0
0167 0003 0030 00F0
0168 0003 0030 00F0
0169 0003 0030 00F0
0170 0003 0030 00F0
0171 0003 0030 00F0
0172 0003 0030 00F0
0173 0003 0030 00F0
0174 0003 0030 00F0
0175 0003 0030 00F0
0176 0003 0030 00F0
0177 0003 0030 00F0
0178 0003 0030 00F0
0179 0003 0030 00F0
0180 0003 0030 00F0
0181 0003 0030 00F0
0182 0003 0030 00F0
0183 0003 0030 00F0
0184 0003 0030 00F0
0185 0003 0030 00F0
0186 00
```



```

0217 4809 46C1 8800 00F0
0218 000C 0000 0000 004C
0219 2809 0000 0000 00F0
021A 4809 AC5C 4030 00F0
021B 4820 0000 0030 00F0

SFT00:
021C 4809 1809 8870 00FC
021D 0000 0000 0C70 00F0
021E 4809 AC55 4030 00F0
021F 4820 0003 0C7C 00F0

SET01:
0220 4809 1800 880C 00F0
0221 000C 0000 0000 001C
0222 4809 AC5C 4030 00FC
0223 4809 1819 880C 00FC
0224 3C00 0C00 003C 0000
0225 4809 AC55 4030 00FC
0226 4820 0019 0070 00F0

CHECK0C:
0227 4809 A0C1 C020 40F0
0228 3030 0003 007C 0040
0229 4809 A156 0B7C 00FC
022A 4809 A001 C070 00FC
022B 4820 0003 0000 00F0

***** END CONDITION CODE ROUTINES *****
CONTENTSRA:
AMPCR = A2
B R = B
4 = SAR; 15 = LIT
LIT AND B = B
REGSTACK - 1 = CPCR
ASE = B
BMR = B
COMP - 16 = SAR
B OR A3 = A3
EINPUL - 1 = CPCR
A2 = AMPCR
JUMP

CONTENTSRRH:
AMPCR = A2
A3 AND LIT = B
15 = LIT

022C 4809 0640 200C 00F0
022D 4809 0C40 883C 00F0
022E 40F0 0000 000C 0080
022F 4809 2C56 0B3C 00FC
0230 0370 000C 0C0C 0060
0231 4809 0C40 883C 24FC
0232 4809 0C40 883C 00F0
0233 4809 0000 0C00 0020
0234 000C 0000 0C00 0020
0235 4809 EC5C 4030 00FC
0236 0380 0003 000C 0060
0237 4809 0C00 0C1C 00F0
0238 4820 6C00 0070 00F0

0239 4809 0640 2C70 00FC
023A 4809 E156 0B7C 00F0
023B 00FC 000C 0070 0060

```


[illegible]

```

0 EOL B
IF FALSE
IF LC2 Y
A3 + 1 =
A3 L = A
COMP 16
A3 R = A
NOT A2 =
A2 + 1 L
A2 R = A
IF LC2 Y
A2 + 1 =

```

[illegible]

138

02E0	032C	00C0	0000	0000	0	INPUT - 1 = CPCR	* READ IN CONTENTS OF A1	01078C00	0
02E1	4809	0040	0000	0000	0	* RESTORE A1	* RESTORE A1	01079C00	0
02E2	0530	0000	0000	0000	0	BHAR 1 = MAR2	* READ IN ADDRESS OF A2	01080C00	0
02E3	0530	0000	0000	0000	0	INPUT - 1 = CPCR	* READ IN A2	01081C00	0
02E4	4809	0040	0000	0000	0	B = A2	* RESTORE A2	01082C00	0
02E5	4809	0040	0000	0000	0	BHAR + 1 = MAR2	* CONSTRUCT ADDRESS OF A3	01083C00	0
02E6	0540	0000	0000	0000	0	INPUT - 1 = CPCR	* READ IN A3	01084C00	0
02E7	4809	0040	0000	0000	0	B = A3	* RESTORE A3	01085C00	0
02E8	4809	0040	0000	0000	0	BHAR + 1 = MAR2	* CREATE ADDR OF MIRC(WORKSPACE + 4)	01086C00	0
02E9	0550	0000	0000	0000	0	INPUT - 1 = CPCR	* READ IN MIRC	01087C00	0
02EA	4809	0040	0000	0000	0	B = MIRC	* RESTORE MIRC	01088C00	0
02EB	4809	0040	0000	0000	0	BHAR + 1 = MAR2	* CREATE ADDR OF RPI	01089C00	0
02EC	0560	0000	0000	0000	0	INPUT - 1 = CPCR	* READ IN RPI	01090C00	0
02ED	4809	0040	0000	0000	0	R L = BR1	* RESTORE RPI	01091C00	0
02EE	0260	0000	0000	0000	0	COMP 8 = SAR J WORKSPACE = LIT	* PUT ADDRESS OF AMPCR IN MAR2	01092C00	0
02EF	4809	0000	0000	0000	0	LIT = MAR2	* READ IN OLD AMPCR	01093C00	0
02F0	0570	0000	0000	0000	0	INPUT - 1 = CPCR	* RESTORE OLD AMPCR	01094C00	0
02F1	4809	0040	0000	0000	0	B = AMPCR	* RESTORE OLD AMPCR	01095C00	0
02F2	4809	0000	0000	0000	0	STEP		01096C00	0
02F3	4820	0000	0000	0000	0	JUMP		01097C00	0
								01098C00	0
								01099C00	0
								01100C00	0
02F4	4809	0000	0000	0000	0	MR2 J IF ROC	* THIS ROUTINE READS THE CONTENT OF THE	01101C00	0
02F5	AC28	0000	0000	0000	0	WHEN ROC THEN BEX J JUMP	* PERFORM READ	01102C00	0
							* READ CONTENTS OF MAR2 INTO B	01103C00	0
								01104C00	0
								01105C00	0
								01106C00	0
								01107C00	0
02F6	9809	0000	0000	1C00	0	MR2 J IF SAI	* THIS ROUTINE WRITES MIRC INTO THE	01108C00	0
02F7	9C28	0000	0000	0000	0	WHEN SAI THEN O J JUMP	* MEMORY ADDRESS OF MAR2	01109C00	0
							* MEMORY WRITE	01110C00	0
							* WHEN WRITE COMPLETED JUMP	01111C00	0
								01112C00	0
								01113C00	0
								01114C00	0
								01115C00	0
								01116C00	0
								01117C00	0
								01118C00	0
								01119C00	0
								01120C00	0
								01121C00	0
								01122C00	0
								01123C00	0
								01124C00	0
								01125C00	0
								01126C00	0
								01127C00	0
								01128C00	0
								01129C00	0
								01130C00	0
								01131C00	0
								01132C00	0
								01133C00	0
								01134C00	0
								01135C00	0
								01136C00	0
								01137C00	0


```

0306 4809 0640 0C40 00F0
0307 4809 20C3 001C 00F0
0308 02C0 00C3 0C40 00F0
0309 2F5C 00C0 0C40 00C0
030A 4809 0000 0000 20F0
030B 4809 0F43 001C 00F0
030C 4809 0000 9000 00F0
030D 0000 00C3 0000 00C0
030E 4809 00C1 1000 00F0
030F 05AC 0000 0000 00C0
0310 4809 0C41 2050 00F0
0311 0000 0000 0000 00F0
0312 4809 0F43 001C 00F0
0313 0500 0000 00C0 00C0
0314 0500 0000 00C0 00C0
0315 4809 0C40 0000 00F0
0316 4809 0F46 0B30 00F0
0317 4809 0000 0000 20F0
0318 4809 0F43 801C 00F0
0319 0000 00C0 0000 001C
031A 4809 0C41 0020 00F0
031B 4809 0000 0000 24F0
031C 4809 0F41 001C 00F0
031D 05C0 0C43 0C00 00C0
031E 4809 0F40 2000 00F0
0320 032C 0003 0C00 00F0
0321 4809 20C3 001C 00F0
0322 2F30 0003 0C00 00C0
0323 4809 0000 0000 20F0
0324 4809 0F43 0C10 00F0
0325 4809 0C41 0020 00F0
0326 0000 00C0 0000 0030
0327 4809 0000 0C00 00F0
0328 4809 0000 0050 00F0
0329 021C 0000 0050 00F0
032A 4809 0C40 4030 00F0
032B 3E53 0C42 0000 00F0
032C 4809 0C42 0000 00F0
032D 5819 0000 0000 00F0
032E 30E0 0000 0000 0000
032F 4809 20C3 001C 00F0
0330 02C0 0003 0000 00F0
0331 2F3C 00C3 0C00 00C0
0332 4809 0C40 0040 00F0
0333 4809 0000 0000 00F0
0334 4820 00C0 0030 00F0

```

```

COMPRESS:
% INPUT OUTPUT CONTROL
% THIS ROUTINE COMPRESSES JNUTK00
% 16 BIT BUFFERS (LHW 16 32 BIT) OVERBOUNDS
% 5-HEBRY WORDS INTO A 32 BIT WORD
% FORMAT COMPATIBLE WITH THE IOP
% UPON ENTRY, B1 CONTAINS BUFFER ADDRESS 01149000
% NO. OF BUFFERS IN UHW OF A3
% THIS ROUTINE ACCEPTS A VARIABLE
% BUFFER LENGTH FOR CRT OUTPUT, J3 WORDS
% STORE AMPCR INTO STACK
% (STACK) = AMPCR
% REFERENCE B1
% STORE BUFFER ADDRESS IN ER2
% CLEAR LHW OF A3 FOR COUNT
% GET CONTENTS OF BUFFER ADDRESS WORD
% STORE CONTENTS IN UHW OF A2
% GET ADJACENT WORD ADDR.
% B = CONTENTS OF ADJACENT WORD
% COMPLEMENT TO NEXT 32 BIT WORD
% STORE NEXT WORD PAIR ADDR. IN B
% REFERENCE B1
% COMPRESSED WORD DESTINATION
% STORE WORD PAIR ADDR. IN B
% REFERENCE BR2
% COMPRESSED WORD DESTINATION IN BR2
% SAVE DESTINATION ADDR. IN B
% BUFFER LENGTH IN 10CW + 2
% 10CW + 2 = HAR2
% B = BUFFER LENGTH
% REFERENCE BR1
% TRANSFER NEXT WORD PAIR ADDR TO BR2
% NEXT DESTINATION ADDR. IN BR1
% INCREMENT COUNTER ( < 2 16 - 1)
% ISOLATE COUNTER IN A2
% SET LC2) SKIP
% RESTORE AMPCR FROM STACK
% SEPARATE AMPCR ASSIGNMENTS AND JUMP
%
%
%

```


[illegible]

THIS ROUTING DUMPS THE CIPHERS


```

0320 2630 00C0 0030 0060
032E 4809 0C40 0030 0060
033F 4809 00C1 0F30 00F0
0340 0C00 00C0 00C0 00F0
0341 4809 0C40 0030 00F0
0342 0680 00C0 00C0 00F0
0343 39AC 00C0 00D0 0040
0344 4809 E0C0 A0C0 00FC
0345 00C0 C0D0 00C0 00FC
0346 4809 E0C0 20C0 00F0
0347 4809 E0C1 90C0 00F0
0348 4809 E0D0 10C0 00F0
0349 48C9 10F1 90D0 00FC
03AA 48C9 C012 00C0 00FC
03AB 6019 00C0 00C0 00F0
03AC 0690 00C0 00C0 0040
03AD 4809 20C3 001C 00F0
03AE 0320 00D0 00C0 00F0
03AF 253C 00C0 00C0 0060
03B0 0809 2641 0020 00F0
03B1 4809 00C0 00C0 00F0
03B2 4809 2640 0030 00F0
03B3 2F50 00C0 C0C0 0060
03B4 3990 00C3 00C0 C040

IN:
03B5 3809 C0C0 00C0 C0F0
03B6 4809 C0C3 A0C0 00F0
03B7 A0D0 00C0 00C0 0010
03B8 48C9 C640 00C0 00F0
03B9 06A0 00C0 0F30 00C0
03BA 48C9 00C0 00C0 00F0
03BB 4824 00C3 00C0 00F0

INDEV:
03BC 0680 00C3 00C0 0040
03BD 06CC 00D0 90D0 0040
03BE 06C0 00C3 00D0 0040

SP01:
03BF 4809 20C3 001C 00F0
03C0 032C 00D0 00C0 00FC
03C1 4809 00C0 00C0 0060
03C2 4809 0C40 0030 00F0
03C3 4809 0C41 0030 00F0
03C4 00C0 00C3 0F00 0030
03C5 48C9 20C1 0F00 00F0
03C6 0060 00C0 00C0 0040
03C7 48C9 0C40 0030 00F0
03C8 06E0 00C3 00C0 0040
03C9 38F0 00C0 00C0 00FC
03CA 48C9 E0C0 90D0 00FC
03CB 00C0 00C0 00C0 0020
03CC 4809 E0C1 1C00 00F0
03CD 4809 E0C1 1000 00F0
03CE 4809 E1C3 1000 00F0
03CF 014C 00C0 00C0 00F0
03D0 334C 00C0 00C0 0060
03D1 4809 E0C3 40D0 00FC

% STORE BUFFER ADDRESS IN IIR
% IS COUNTER = 0
% REGENERATE BUFFER ADDRESS
% WRITE NEW ADDRESS IN BR1
% WRITE NEW BUFFER ADDR IN I0C4 + 1
% THIS ROUTINE INPUTS DATA FROM
% PERIPHERAL DEVICES TO THE DESIGNATED
% BUFFERS. CRO READER, CRT, OR DISK (N1)
% CLEAR LC2
% A2 = (I0C4)
% A2 CONTAINS DEVICE CODE
%
% DISK NOT IMPLEMENTED
% READ SP0 FUNCTION:
% RETRIEVE BUFFER ADDR.
% STORE BUFFER ADDR IN IIR
% STORE ADDRESS IN BR1
% ISOLATE COUNTER
% DECREMENT COUNTER
% RESTORE A3, LHM CLEARED
% BUFFER EXPANSION FACTOR IN LHM OF A3
% EXPAND BUFFER
% ISOLATE COUNTER

```



```

C402 4809 2003 001C 00F0
C403 02CC 0000 0000 00E0
C404 4809 E000 0000 00F0
C405 2F5C 00C3 0030 00E0
C406 0720 0003 00C0 00E0
C407 0000 0000 0000 00E0
C408 0000 0000 0000 00E0
C409 4809 0000 9C00 00F0
C40A 4809 0C41 0020 00F0
C40B 4809 LC5C 1000 00F0
C40C 238C 00C3 0070 00E0
C40D 4809 E000 AC0C 00F0
C40E 0000 0000 0070 00E0
C40F 4809 CC40 0030 00F0
C410 0730 0000 0070 00E0

C411 4809 2003 001C 00F0
C412 02CC 0003 0030 00E0
C413 4809 E000 0030 00F0
C414 2F5C 00C0 00C0 00E0
C415 074C 00C3 0000 00E0
C416 0750 00C0 00C0 00E0

C417 4809 E001 1070 00F0
C418 0000 0000 00C0 00E0
C419 4809 E003 3070 00F0
C41A 4809 0C41 0010 00F0
C41B 0000 00C0 0000 003C
C41C 4809 0C41 003C 00F0
C41D 0000 0000 0000 0020
C41E 4809 EC5C 1070 00F0
C41F 076C 00C0 0070 00E0
C420 4809 CC40 AC0C 00F0
C421 400C 00C0 0030 0010
C422 4809 00C3 0070 00E0
C423 5600 0000 0000 00F0
C424 077C 0000 0000 009C
C425 0780 00C3 0070 00E0

C426 4809 18C1 AC0C 00F0
C427 3000 00C3 0030 00E0
C428 4809 CC5C AC0C 00F0
C429 300C 0000 0070 001C

I1W1 AT Y = IY1 + (RCH)
% LC2 INDICATES BYTE INSTRUCTION
% ROUTINES RETURN B = Y

% STORE A3 WITH RETURN ADDR. IN STACK
% WRITE TO EMULATION RESERVE BUFFER
% CLEAR UHW OF A3
% UHW OF A3 CONTAINS IY1
% LHW OF A3 CONTAINS INSTRUCTION
% B = (RCH)
% A2 CONTAINS IY1
% CALCULATE ADDRESS OF Y
%
% INDIRECT ADDRESSING WITHOUT INDEXING
% IY AT Y = IY1
% LC2 INDICATES BYTE INST. ROUTINE
% RETURNS B = Y AND A2 = (RCH)
% WRITE A3 WITH RETURN IN UHW
% RETRIEVE IY1 FIELD
%
% THIS ROUTINE READS IN IY1 AND TESTS
% IF CASCADED OR DIRECT ADDRESSING IS
% REQUIRED.
% CLEAR UHW OF A3
%
% U2 = IY1 ADDRESS
% SAVE IY1 ADDR IN UHW OF A3
% B = (IY1)
% TEST IF BIT 14 OF IY1 IS SET
% IF BIT SET, CASCADED IY
% IF BIT NOT SET, FORM Y
%
% THIS ROUTINE DECIDES WHICH ADDRESSING
% FORMULA TO USE TO COMPUTE THE NEW
% LOCATION OF THE INDIRECT WORD
% A2 IS BIT MASK FOR J FIELD OF IY

```


042A	4809	C640	003C	00F0	A2 + AMPCR = AMPCR	% COMPUTE FORMULA DESIRED	0148FC00 D
042B	079C	0003	003C	C0C0	INADDR - 1 = AMPCR		01499F00 D
042C	4809	0000	0030	00F0	STEP		015004C0 D
042D	4824	0000	003C	00F0	EXEC		01502000 D
042E	07A0	0000	003C	0040			01503100 D
042F	07A0	0000	003C	0040	AD10 - 1 = MPCR		01504000 D
0430	07A0	0000	003C	0040	AD11 - 1 = MPCR		01505100 D
0431	07A0	0000	003C	0040	AD12 - 1 = MPCR		01506100 D
0432	07A0	0000	003C	0040	AD13 - 1 = MPCR		01507000 D
0433	4809	E0D0	A03C	00F0			01508000 D
0434	000C	0000	0030	0020	A3 R = A2	% NEW 1W AT Y = 1W2	01509000 D
0435	000C	0003	0030	00F0	16 = SAR	% SHIFT ADDR OF 1W INTO A2	01510000 D
0436	070C	0000	0030	003C	A2 OR 1 L = BR2	% NEXT ADDRESS OF 1W	01511000 D
0437	070C	0000	0030	003C	COMP 8 = SAR		01512000 D
0438	000C	0000	0030	003C	ENULIN - 1 = CPCR	% R CONTAINS 1W2; Y = 1W2	01513000 D
0439	416C	00C0	0030	0040	INDIRMO - 1 = MPCR	% Y = NEXT ADDR OF 1W1	01514000 D
043A	000C	0000	0030	003C			01515000 D
043B	000C	0000	0030	003C			01516000 D
043C	000C	0000	0030	003C			01517000 D
043D	000C	0000	0030	003C			01518000 D
043E	000C	0000	0030	003C			01519000 D
043F	000C	0000	0030	003C			01520000 D
0440	000C	0000	0030	003C			01521000 D
0441	000C	0000	0030	003C			01522000 D
0442	000C	0000	0030	003C			01523000 D
0443	000C	0000	0030	003C			01524000 D
0444	000C	0000	0030	003C			01525000 D
0445	000C	0000	0030	003C			01526000 D
0446	000C	0000	0030	003C			01527000 D
0447	000C	0000	0030	003C			01528000 D
0448	000C	0000	0030	003C			01529000 D
0449	000C	0000	0030	003C			01530000 D
044A	000C	0000	0030	003C			01531000 D
044B	000C	0000	0030	003C			01532000 D
044C	000C	0000	0030	003C			01533000 D
044D	000C	0000	0030	003C			01534000 D
044E	000C	0000	0030	003C			01535000 D
044F	000C	0000	0030	003C			01536000 D
0450	000C	0000	0030	003C			01537000 D
0451	000C	0000	0030	003C			01538000 D
0452	000C	0000	0030	003C			01539000 D
0453	000C	0000	0030	003C			01540000 D
0454	000C	0000	0030	003C			01541000 D
0455	000C	0000	0030	003C			01542000 D
0456	000C	0000	0030	003C			01543000 D
0457	000C	0000	0030	003C			01544000 D
0458	000C	0000	0030	003C			01545000 D
0459	000C	0000	0030	003C			01546000 D
045A	000C	0000	0030	003C			01547000 D
045B	000C	0000	0030	003C			01548000 D
045C	000C	0000	0030	003C			01549000 D
045D	000C	0000	0030	003C			01550000 D
045E	000C	0000	0030	003C			01551000 D
045F	000C	0000	0030	003C			01552000 D
0460	000C	0000	0030	003C			01553000 D
0461	000C	0000	0030	003C			01554000 D
0462	000C	0000	0030	003C			01555000 D
0463	000C	0000	0030	003C			01556000 D
0464	000C	0000	0030	003C			01557000 D

00A3	4809	C640	2000	00F0	AMPCR = A2	% PAGE ADDRESSING IS ALLOWED	01679C00
00A4	4809	0000	0000	20F0	ASR	% SAVE RETURN IN A2	01679C00
00A5	4809	0F43	001F	00F0	BHAR = DR2	% BR1 = BR2	01681000
00A6	0970	0000	0000	00F0	EMULIN - 1 = CPCR	% G = (BR1)	01681000
00A7	4809	0C40	0C30	00F0	R = MIR	% TRANSFER B TO MIR	01682C00
00A8	4809	0C00	0000	00F0	A2 = B	% STORE AMPCR IN B	01683C00
00A9	4809	E001	0C10	00F0	A3 L = BR2	% LHM OF A3 INTO BR2	01684C00
00AA	0C00	0000	0000	00F0	COMP B = SAR	% LHM OF A3 INTO BR2	01685C00
00AB	4809	E000	9C00	00F0	A3 R = A3	% ZERO OUT LHM OF A3	01686C00
00AC	0000	0000	0000	00F0	16 = SAR	% ZERO OUT LHM OF A3	01687C00
00AD	4809	E0C1	1C00	00F0	A3 L = A3	% STORE AMPCR IN LHM OF A3	01688C00
00AE	4809	E5C0	1000	00F0	A3 OR B = A3	% STORE AMPCR IN LHM OF A3	01689C00
00AF	0980	0000	9000	00F0	EMULOUT - 1 = CPCR	% STORE AMPCR IN LHM OF A3	01690C00
00B0	4809	E0F1	2000	00F0	A3 L = A2	% STORE AMPCR IN LHM OF A3	01691C00
00B1	0000	0000	0000	00F0	COMP 16 = SAR	% RETURN AMPCR TO LHM OF A2	01692C00
00B2	4809	C0C0	AC00	00F0	A2 R = A2	% ZERO LHM OF A3	01693C00
00B3	4809	E0C0	9000	00F0	A3 R = A3	% RETURN AMPCR TO LHM OF A2	01694C00
00B4	4809	E0C1	1000	00F0	A3 L = A3	% ZERO LHM OF A3	01695C00
00B5	4809	E5C0	1000	00F0	BHAR = B	% INCREMENT ADDRESS DESTINATION	01696C00
00B6	4809	E5C0	1000	00F0	AMPCR = A3, ASR	% STORE DESTINATION ADDR IN LHM OF A3	01697C00
00B7	4809	0C40	8000	00F0	BHAR = B, CEAR	% INCREMENT ORIGIN	01698C00
00B8	0000	0000	0000	00F0	B = SAR	% INCREMENT ORIGIN	01699C00
00B9	4809	0C40	0000	00F0	B + 1 L = PR1	% SHIFT COUNT TO LHM OF A3	01700C00
00BA	4809	E001	9C00	00F0	A3 C = A3	% DECREMENT COUNTER	01701C00
00BB	0000	0000	0000	00F0	16 = SAR	% ISOLATE COUNTER IN LHM OF B	01702C00
00BC	4809	E30E	1B00	00F0	A3 - 1 = A3, B	% RESTORE A3	01703C00
00BD	4809	0C41	0B00	00F0	B L = B	% RESTORE A3	01704C00
00BE	4809	0C41	0B00	00F0	E R = B	% RESTORE A3	01705C00
00BF	4809	E0F1	9000	00F0	A3 C = A3	% RESTORE A3	01706C00
00C0	4809	0C52	0C00	00F0	R EOL 0	% RESTORE A3	01707C00
00C1	0000	0000	0000	00F0	COMP B = SAR	% RESTORE A3	01708C00
00C2	6000	0000	0000	00F0	IF FALSE THEN STEP ELSE SKIP	% RESTORE A3	01709C00
00C3	4A3C	0000	0000	00F0	HRPT - 1 = MPCR	% RESTORE A3	01710C00
00C4	4809	C0C0	0F40	00F0	A2 = AMPCR	% RESTORE A3	01711C00
00C5	48C9	C0C0	0C00	00F0	STEP	% RESTORE A3	01712C00
00C6	482D	0000	0000	00F0	JUMP	% RESTORE A3	01713C00
00C7	4809	0000	1000	00F0	C = A3	% RESTORE A3	01714C00
00C8	4809	0000	0000	00F0	16 = SAR	% RESTORE A3	01715C00
00C9	2009	0000	0000	00F0	IF LCI THEN SET LCI, SKIP	% RESTORE A3	01716C00
00CA	0990	0000	0000	00F0	REG - 1 = MPCR	% RESTORE A3	01717C00
00CB	4809	0C52	0C00	00F0	IF FALSE THEN A2 XOR R, STEP ELSE JUMP	% RESTORE A3	01718C00
00CC	5400	C040	0000	00F0	A2, IF M4T THEN SET LC2, FLAG FOR NEG PRODUCT	% RESTORE A3	01719C00
00CD	4409	C040	0000	00F0	IF M4T THEN NOT A2 = A2, STEP ELSE SKIP	% RESTORE A3	01720C00
00CE	4C0B	C0C2	2000	00F0	IF M4T THEN NOT A2 = A2, STEP ELSE SKIP	% RESTORE A3	01721C00
00CF	48C9	C0C0	2000	00F0	A2 + 1 = A2	% RESTORE A3	01722C00
00D0	4809	0C40	0C00	00F0	P	% RESTORE A3	01723C00
00D1	4809	0C5E	0B00	00F0	IF M4T THEN 0 - B = B, COMPLEMENT F	% RESTORE A3	01724C00
00D2	2B19	C0C0	0000	00F0	IF LCI THEN SKIP	% RESTORE A3	01725C00
00D3	09AC	0000	0000	00F0	REG - 1 = MPCR	% RESTORE A3	01726C00
00D4	48C9	C0C0	0000	00F0	A2	% RESTORE A3	01727C00
00D5	5C09	EC40	1000	00F0	IF LST THEN A3 + B = A3	% RESTORE A3	01728C00
00D6	48C9	C0C0	4000	00F0	A2 R = A2, CSAR	% RESTORE A3	01729C00
00D7	1000	C0C0	0000	00F0	1 = SAR	% RESTORE A3	01730C00


```

0502 0440 0000 0000 0040      * NEG SUM, LIKE SIGNS, POS X
                                * 01796000 0

0503 2808 0000 0000 0000      * IF LC1 THEN STEP ELSE SKIP * THIS ROUTINE WILL SET THE
                                * 01801000 0
0504 0450 0000 0000 0040      * SETIOBIT - 1 = MPCR
                                * 01800000 0
0505 0460 0000 0000 0040      * CLEAROV - 1 = MPCR
                                * 01802000 0
                                * 01803000 0
                                * 01804000 0
                                * 01805000 0
                                * 01806000 0
                                * 01807000 0
                                * 01808000 0
                                * 01809000 0
                                * 01810000 0
                                * 01811000 0
                                * 01812000 0
                                * 01813000 0
                                * 01814000 0
                                * 01815000 0
                                * 01816000 0
                                * 01817000 0
                                * 01818000 0
                                * 01819000 0
                                * 01820000 0
                                * 01821000 0
                                * 01822000 0
                                * 01823000 0
                                * 01824000 0
                                * 01825000 0
                                * 01826000 0
                                * 01827000 0
                                * 01828000 0
                                * 01829000 0
                                * 01830000 0
                                * 01831000 0
                                * 01832000 0
                                * 01833000 0
                                * 01834000 0
                                * 01835000 0
                                * 01836000 0
                                * 01837000 0
                                * 01838000 0
                                * 01839000 0
                                * 01840000 0
                                * 01841000 0
                                * 01842000 0
                                * 01843000 0
                                * 01844000 0
                                * 01845000 0
                                * 01846000 0
                                * 01847000 0
                                * 01848000 0
                                * 01849000 0
                                * 01850000 0
                                * 01851000 0
                                * 01852000 0
                                * 01853000 0
                                * 01854000 0
                                * 01855000 0
                                * 01856000 0
                                * 01857000 0

0506 3000 0001 0000 0000      * THIS ROUTINE WILL CLEAR THE OVERFLOW
                                * 01860000 0
0507 3000 0000 0000 0000      * 27 = SAR
                                * 01861000 0
0508 4809 0000 0000 0000      * A1 AND R110 = A1,CSAR
                                * 01862000 0
0509 2809 0001 0000 0000      * A1 C = A1; IF LC1
                                * 01863000 0
050A 4820 0000 0000 0000      * JUMP
                                * 01864000 0
                                * 01865000 0
                                * 01866000 0
                                * 01867000 0
                                * 01868000 0
                                * 01869000 0
                                * 01870000 0
                                * 01871000 0
                                * 01872000 0
                                * 01873000 0
                                * 01874000 0
                                * 01875000 0
                                * 01876000 0
                                * 01877000 0
                                * 01878000 0
                                * 01879000 0
                                * 01880000 0
                                * 01881000 0
                                * 01882000 0
                                * 01883000 0
                                * 01884000 0
                                * 01885000 0
                                * 01886000 0
                                * 01887000 0
                                * 01888000 0
                                * 01889000 0
                                * 01890000 0
                                * 01891000 0
                                * 01892000 0
                                * 01893000 0
                                * 01894000 0
                                * 01895000 0
                                * 01896000 0
                                * 01897000 0
                                * 01898000 0
                                * 01899000 0
                                * 01900000 0

050B 4809 0001 0000 0000      * SETIOBIT:
                                * 01901000 0
                                * 01902000 0
                                * 01903000 0
                                * 01904000 0
                                * 01905000 0
                                * 01906000 0
                                * 01907000 0
                                * 01908000 0
                                * 01909000 0
                                * 01910000 0
                                * 01911000 0
                                * 01912000 0
                                * 01913000 0
                                * 01914000 0
                                * 01915000 0
                                * 01916000 0
                                * 01917000 0
                                * 01918000 0
                                * 01919000 0
                                * 01920000 0
                                * 01921000 0
                                * 01922000 0
                                * 01923000 0
                                * 01924000 0
                                * 01925000 0
                                * 01926000 0
                                * 01927000 0
                                * 01928000 0
                                * 01929000 0
                                * 01930000 0
                                * 01931000 0
                                * 01932000 0
                                * 01933000 0
                                * 01934000 0
                                * 01935000 0
                                * 01936000 0
                                * 01937000 0
                                * 01938000 0
                                * 01939000 0
                                * 01940000 0
                                * 01941000 0
                                * 01942000 0
                                * 01943000 0
                                * 01944000 0
                                * 01945000 0
                                * 01946000 0
                                * 01947000 0
                                * 01948000 0
                                * 01949000 0
                                * 01950000 0
                                * 01951000 0
                                * 01952000 0
                                * 01953000 0
                                * 01954000 0
                                * 01955000 0
                                * 01956000 0
                                * 01957000 0
                                * 01958000 0
                                * 01959000 0
                                * 01960000 0
                                * 01961000 0
                                * 01962000 0
                                * 01963000 0
                                * 01964000 0
                                * 01965000 0
                                * 01966000 0
                                * 01967000 0
                                * 01968000 0
                                * 01969000 0
                                * 01970000 0
                                * 01971000 0
                                * 01972000 0
                                * 01973000 0
                                * 01974000 0
                                * 01975000 0
                                * 01976000 0
                                * 01977000 0
                                * 01978000 0
                                * 01979000 0
                                * 01980000 0
                                * 01981000 0
                                * 01982000 0
                                * 01983000 0
                                * 01984000 0
                                * 01985000 0
                                * 01986000 0
                                * 01987000 0
                                * 01988000 0
                                * 01989000 0
                                * 01990000 0
                                * 01991000 0
                                * 01992000 0
                                * 01993000 0
                                * 01994000 0
                                * 01995000 0
                                * 01996000 0
                                * 01997000 0
                                * 01998000 0
                                * 01999000 0
                                * 02000000 0

050C 4809 0000 0000 0000      * PRIERR:
                                * 02001000 0
                                * 02002000 0
                                * 02003000 0
                                * 02004000 0
                                * 02005000 0
                                * 02006000 0
                                * 02007000 0
                                * 02008000 0
                                * 02009000 0
                                * 02010000 0
                                * 02011000 0
                                * 02012000 0
                                * 02013000 0
                                * 02014000 0
                                * 02015000 0
                                * 02016000 0
                                * 02017000 0
                                * 02018000 0
                                * 02019000 0
                                * 02020000 0
                                * 02021000 0
                                * 02022000 0
                                * 02023000 0
                                * 02024000 0
                                * 02025000 0
                                * 02026000 0
                                * 02027000 0
                                * 02028000 0
                                * 02029000 0
                                * 02030000 0
                                * 02031000 0
                                * 02032000 0
                                * 02033000 0
                                * 02034000 0
                                * 02035000 0
                                * 02036000 0
                                * 02037000 0
                                * 02038000 0
                                * 02039000 0
                                * 02040000 0
                                * 02041000 0
                                * 02042000 0
                                * 02043000 0
                                * 02044000 0
                                * 02045000 0
                                * 02046000 0
                                * 02047000 0
                                * 02048000 0
                                * 02049000 0
                                * 02050000 0
                                * 02051000 0
                                * 02052000 0
                                * 02053000 0
                                * 02054000 0
                                * 02055000 0
                                * 02056000 0
                                * 02057000 0
                                * 02058000 0
                                * 02059000 0
                                * 02060000 0
                                * 02061000 0
                                * 02062000 0
                                * 02063000 0
                                * 02064000 0
                                * 02065000 0
                                * 02066000 0
                                * 02067000 0
                                * 02068000 0
                                * 02069000 0
                                * 02070000 0
                                * 02071000 0
                                * 02072000 0
                                * 02073000 0
                                * 02074000 0
                                * 02075000 0
                                * 02076000 0
                                * 02077000 0
                                * 02078000 0
                                * 02079000 0
                                * 02080000 0
                                * 02081000 0
                                * 02082000 0
                                * 02083000 0
                                * 02084000 0
                                * 02085000 0
                                * 02086000 0
                                * 02087000 0
                                * 02088000 0
                                * 02089000 0
                                * 02090000 0
                                * 02091000 0
                                * 02092000 0
                                * 02093000 0
                                * 02094000 0
                                * 02095000 0
                                * 02096000 0
                                * 02097000 0
                                * 02098000 0
                                * 02099000 0
                                * 02100000 0

050D 4809 0000 0000 0000      * CLEAROV - 1 = CPCR
                                * 02101000 0
050E 0470 0000 0000 0000      * WRITE A BLANK LINE
                                * 02102000 0
050F 4809 0000 0000 0000      * AMPCR = A3
                                * 02103000 0
                                * 02104000 0
                                * 02105000 0
                                * 02106000 0
                                * 02107000 0
                                * 02108000 0
                                * 02109000 0
                                * 02110000 0
                                * 02111000 0
                                * 02112000 0
                                * 02113000 0
                                * 02114000 0
                                * 02115000 0
                                * 02116000 0
                                * 02117000 0
                                * 02118000 0
                                * 02119000 0
                                * 02120000 0
                                * 02121000 0
                                * 02122000 0
                                * 02123000 0
                                * 02124000 0
                                * 02125000 0
                                * 02126000 0
                                * 02127000 0
                                * 02128000 0
                                * 02129000 0
                                * 02130000 0
                                * 02131000 0
                                * 02132000 0
                                * 02133000 0
                                * 02134000 0
                                * 02135000 0
                                * 02136000 0
                                * 02137000 0
                                * 02138000 0
                                * 02139000 0
                                * 02140000 0
                                * 02141000 0
                                * 02142000 0
                                * 02143000 0
                                * 02144000 0
                                * 02145000 0
                                * 02146000 0
                                * 02147000 0
                                * 02148000 0
                                * 02149000 0
                                * 02150000 0
                                * 02151000 0
                                * 02152000 0
                                * 02153000 0
                                * 02154000 0
                                * 02155000 0
                                * 02156000 0
                                * 02157000 0
                                * 02158000 0
                                * 02159000 0
                                * 02160000 0
                                * 02161000 0
                                * 02162000 0
                                * 02163000 0
                                * 02164000 0
                                * 02165000 0
                                * 02166000 0
                                * 02167000 0
                                * 02168000 0
                                * 02169000 0
                                * 02170000 0
                                * 02171000 0
                                * 02172000 0
                                * 02173000 0
                                * 02174000 0
                                * 02175000 0
                                * 02176000 0
                                * 02177000 0
                                * 02178000 0
                                * 02179000 0
                                * 02180000 0
                                * 02181000 0
                                * 02182000 0
                                * 02183000 0
                                * 02184000 0
                                * 02185000 0
                                * 02186000 0
                                * 02187000 0
                                * 02188000 0
                                * 02189000 0
                                * 02190000 0
                                * 02191000 0
                                * 02192000 0
                                * 02193000 0
                                * 02194000 0
                                * 02195000 0
                                * 02196000 0
                                * 02197000 0
                                * 02198000 0
                                * 02199000 0
                                * 02200000 0

0510 4F40 0000 0000 0000      * LIT L = B
                                * 02201000 0
0511 0470 0000 0000 0000      * 8 = LIT; COMP 16 = SAR
                                * 02202000 0
0512 4809 0000 0000 0000      * A3 OR B = A3
                                * 02203000 0
0513 0590 0000 0000 0000      * MOVE - 1 = CPCR
                                * 02204000 0
0514 4809 2001 0000 0000      * WRITE ERROR LIST MESSAGE
                                * 02205000 0
0515 0080 0000 0000 0000      * CLEAROV - 1 = CPCR
                                * 02206000 0
0516 4809 EC50 1000 0000      * WRITEBUFF - 1 = CPCR
                                * 02207000 0
0517 4420 0000 0000 0000      * CLEAROV - 1 = CPCR
                                * 02208000 0
0518 0480 0000 0000 0000      * RESTORE LU REGISTERS
                                * 02209000 0
0519 4F40 0000 0000 0000      * STOPTIME - 1 = MPCR
                                * 02210000 0
051A 04AC 0000 0000 0000      *
                                * 02211000 0
051B 0480 0000 0000 0000      *
                                * 02212000 0
051C 0480 0000 0000 0000      *
                                * 02213000 0
                                * 02214000 0
                                * 02215000 0
                                * 02216000 0
                                * 02217000 0
                                * 02218000 0
                                * 02219000 0
                                * 02220000 0
                                * 02221000 0
                                * 02222000 0
                                * 02223000 0
                                * 02224000 0
                                * 02225000 0
                                * 02226000 0
                                * 02227000 0
                                * 02228000 0
                                * 02229000 0
                                * 02230000 0
                                * 02231000 0
                                * 02232000 0
                                * 02233000 0
                                * 02234000 0
                                * 02235000 0
                                * 02236000 0
                                * 02237000 0
                                * 02238000 0
                                * 02239000 0
                                * 02240000 0
                                * 02241000 0
                                * 02242000 0
                                * 02243000 0
                                * 02244000 0
                                * 02245000 0
                                * 02246000 0
                                * 02247000 0
                                * 02248000 0
                                * 02249000 0
                                * 02250000 0
                                * 02251000 0
                                * 02252000 0
                                * 02253000 0
                                * 02254000 0
                                * 02255000 0
                                * 02256000 0
                                * 02257000 0
                                * 02258000 0
                                * 02259000 0
                                * 02260000 0
                                * 02261000 0
                                * 02262000 0
                                * 02263000 0
                                * 02264000 0
                                * 02265000 0
                                * 02266000 0
                                * 02267000 0
                                * 02268000 0
                                * 02269000 0
                                * 02270000 0
                                * 02271000 0
                                * 02272000 0
                                * 02273000 0
                                * 02274000 0
                                * 02275000 0
                                * 02276000 0
                                * 02277000 0
                                * 02278000 0
                                * 02279000 0
                                * 02280000 0
                                * 02281000 0
                                * 02282000 0
                                * 02283000 0
                                * 02284000 0
                                * 02285000 0
                                * 02286000 0
                                * 02287000 0
                                * 02288000 0
                                * 02289000 0
                                * 02290000 0
                                * 02291000 0
                                * 02292000 0
                                * 02293000 0
                                * 02294000 0
                                * 02295000 0
                                * 02296000 0
                                * 02297000 0
                                * 02298000 0
                                * 02299000 0
                                * 02300000 0

051D 4809 0C43 0010 0000      * THIS ROUTINE CHECKS THE STACK
                                * 02301000 0
051E 4809 A001 0000 0000      * DESIGNATION BIT IN SR01 TO DETERMINE
                                * 02302000 0
051F 0000 0000 0000 0000      * WHICH STACK THE PROGRAMMER IS USING.
                                * 02303000 0
0520 4809 A000 0000 0000      * REGISTER IS ASSUMED TO HAVE THE
                                * 02304000 0
0521 5800 0000 0000 0000      * KEYWORD.
                                * 02305000 0
0522 04CC 0000 0000 0000      * WRAPAROUND IS ALLOWED.
                                * 02306000 0
0523 4809 2C52 0000 0000      * STORE THE REGISTER
                                * 02307000 0
0524 0100 0000 0000 0000      * POSITION STACK DESIGNATOR BIT IN LS
                                * 02308000 0
0525 5800 0000 0000 0000      * POSITION REG BIT. IN LS BIT
                                * 02309000 0
0526 4809 0C03 0010 0000      * SEND A1 TO THE ADDR
                                * 02310000 0
0527 4820 A001 0000 0000      * IF LAST THEN STEP ELSE SKIP
                                * 02311000 0
                                * 02312000 0
                                * 02313000 0
                                * 02314000 0
                                * 02315000 0
                                * 02316000 0
                                * 02317000 0
                                * 02318000 0
                                * 02319000 0
                                * 02320000 0
                                * 02321000 0
                                * 02322000 0
                                * 02323000 0
                                * 02324000 0
                                * 02325000 0
                                * 02326000 0
                                * 02327000 0
                                * 02328000 0
                                * 02329000 0
                                * 02330000 0
                                * 02331000 0
                                * 02332000 0
                                * 02333000 0
                                * 02334000 0
                                * 02335000 0
                                * 02336000 0
                                * 02337000 0
                                * 02338000 0
                                * 02339000 0
                                * 02340000 0
                                * 02341000 0
                                * 02342000 0
                                * 02343000 0
                                * 02344000 0
                                * 02345000 0
                                * 02346000 0
                                * 02347000 0
                                * 02348000 0
                                * 02349000 0
                                * 02350000 0
                                * 02351000 0
                                * 02352000 0
                                * 02353000 0
                                * 02354000 0
                                * 02355000 0
                                * 02356000 0
                                * 02357000 0
                                * 02358000 0
                                * 02359000 0
                                * 02360000 0
                                * 02361000 0
                                * 02362000 0
                                * 02363000 0
                                * 02364000 0
                                * 02365000 0
                                * 02366000 0
                                * 02367000 0
                                * 02368000 0
                                * 02369000 0
                                * 02370000 0
                                * 02371000 0
                                * 02372000 0
                                * 02373000 0
                                * 02374000 0
                                * 02375000 0
                                * 02376000 0
                                * 02377000 0
                                * 02378000 0
                                * 02379000 0
                                * 02380000 0
                                * 02381000 0
                                * 02382000 0
                                * 02383000 0
                                * 02384000 0
                                * 02385000 0
                                * 02386000 0
                                * 02387000 0
                                * 02388000 0
                                * 02389000 0
                                * 02390000 0
                                * 02391000 0
                                * 02392000 0
                                * 02393000 0
                                * 02394000 0
                                * 02395000 0
                                * 02396000 0
                                * 02397000 0
                                * 02398000 0
                                * 02399000 0
                                * 02400000 0

```



```

0024 8809 2C32 001C 00F0
0529 0100 0030 00F0 00E0
052A 8809 2C32 0000 00F0
052B 0200 0000 0030 00E0
052C 6819 0009 0030 00F0
052D 4809 A0C1 0000 00F0
052E 8809 20C3 001C 00F0
052F 0100 0000 0000 00E0
0530 1820 A0C1 0000 00F0

0531 8809 0640 0030 00F0
0532 8809 20C3 001C 00F0
0533 0200 0000 0030 00E0
0534 2F30 0009 0000 00E0
0535 8809 0C40 0030 00F0
0536 2F30 0000 0030 00E0
0537 2F30 0000 0030 00E0
0538 8809 0C40 2000 00F0
0539 8809 0F45 001C 00F0
053A 2F30 0003 0000 00E0
053B 8809 0C40 1000 00F0
053C 8809 20C3 001C 00F0
053D 2F30 0000 0030 00E0
053E 8809 0C40 0030 00F0
053F 8809 0C40 0030 00F0
0540 8809 0003 0030 00F0
0541 1820 0003 0000 00F0

0542 8809 0640 0030 00F0
0543 8809 2003 001C 00F0
0544 0200 0000 0000 00E0
0545 2F30 0000 0000 00E0
0546 8809 0C40 2000 00F0
0547 0030 0003 0000 00E0
0548 0030 0003 0000 00E0
0549 8809 C0C0 A030 00F0
054A 8809 C131 2030 00F0
054B 8809 C131 2030 00F0
054C 8809 A0C3 1000 00F0
054D 8809 A0C1 4030 00F0
054E 8809 AC5C 4030 00F0
054F 8809 880E 4000 00F0
0550 0A00 0003 0030 00F0
0551 8809 C0C0 0030 00F0
0552 8809 A000 C0C0 00F0
0553 0000 0000 0030 00E0
0554 8809 A0C1 4030 00F0
0555 8809 E0C3 8030 00F0
0556 8809 AC5C 4000 00F0
0557 8809 E0C3 0030 00F0
0558 8809 0000 0000 00E0
0559 883F 0000 0000 00F0

LIT + U = MARZ
16 = LIT
LIT EOL B
32 = LIT
IF TRUE THEN SKIP
A1 C = A1 J JUMP
LIT = MAR2
16 = LIT
A1 C = A1 J JUMP

AMPCR = MIR
LIT + 1 = MAR2
WORKSPACE = LIT
EINPUT - 1 = CPCR
B = A1
EINPUT + 1 = MAR2
EINPUT - 1 = CPCR
EINPUT + 1 = MAR2
EINPUT - 1 = CPCR
R = A3
LIT = MAR2
EINPUT - 1 = CPCR
P = MIR+BHI
B = AMPCR+BHI
STEP
JUMP

AMPCR = MIR
LIT = MAR2
PSW = LIT
EINPUT - 1 = CPCR
B = MIR+BHI
LIT = A2
LIT + 2 = LIT
A2 R = A2
A2 + LIT L = A2
A2 OR B = A3
A1 R = A1
A1 L = A1, BHI
A1 OR B = A1
A1 AND B011 = A1
IFETCH - 1 = CPCR
B = MIR
A1 R = A1
16 = SAR
A1 L = A1
A3 R = B
A3 OR B = A1+BHI
A3 = AMPCR
STEP
RETN

% THIS ROUTINE RESTORES THE REGISTERS
% OF THE LU IN WORKSPACE
% SAVE RETURN IN MIR
% MAR2 = WORKSPACE + 1

% RESTORE A1
% INCREMENT WORKSPACE ADDR
% RESTORE A2
% INCREMENT WORKSPACE ADDR
% RESTORE A3
% SELECT WORKSPACE BASE ADDR
% SWAP B AND MIR
% RESTORE AMPCR AND B

% THIS ROUTINE WILL PLACE THE CONTENTS
% OF PSW INTO THE PAR, THEN RESTORE
% THE PAR TO (PAR) + 2.
% SAVE RETURN ADDR TO OPCCOE IN MIR
% ADDRESS OF PSW

% (PSW) INTO B
% (PSW) INTO MIR, RETURN ADDR INTO 9
% PAR INTO A2
% RIGHT JUSTIFY (PAR)
% (PAR) + 2 = A2
% (PAR)+2/RETURN ADDR = A3
% CLEAR PAR

% (PSW) INTO PAR
% CLEAR BIASED FETCH BIT
% INSTRUCTION AT Y INTO B
% CLEAR PAR

% (PAR) + 2 = F
% RESTORE RETURN ADDR, INSTR IN B

```



```

055A 4809 0000 0000 0000
055B 4809 EC01 1C00 4000
055C 3000 0000 0000 0000
055D 4809 0000 0000 0000
055E 4809 0000 0000 0000
055F 4809 0000 0000 0000
0560 4809 EC01 1C00 4000
0561 0000 0000 0000 0000
0562 4809 0000 0000 0000
0563 4809 EC01 1C00 4000
0564 4809 0000 0000 0000
0565 4809 0000 0000 0000
0566 4809 AC00 0000 0000
0567 0AEC 0000 0000 0000

0568 4809 0000 0000 2400
0569 4809 0000 0000 0000
056A 4809 0000 0000 0000
056B 51C0 0000 0000 0000

056C 2F30 0000 0000 0000
056D 4809 0000 0000 0000
056E 4809 0000 0000 0000
056F 4809 0000 0000 0000
0570 4820 0000 0000 0000

0571 4809 E001 9000 0000
0572 0000 0000 0000 0000
0573 4809 EC01 1C00 4000
0574 4809 0000 0000 0000
0575 4809 EC01 1C00 4000
0576 0000 0000 0000 0000
0577 4809 0000 0000 0000
0578 4809 0000 0000 0000
0579 4809 0000 0000 0000
057A 4809 2500 0000 0000
057B 0000 0000 0000 0000
057C 7800 0000 0000 0000
057D 0000 0000 0000 0000
057E 4809 0000 2000 0000
057F 5000 0000 0000 0000
0580 381C 0000 0000 0000

X THIS ROUTINE COMPUTES (P)*D = P
X AND JUMPS TO NEXT INSTRUCTION
X EXAMINE SIGN BIT OF *D*
X A3 TO THE ADDR
IF HST THEN B111 L = B
X IF SIGN=1, FILL UPPER BYTE WITH 1'S
X RESTORE *D* FIELD TO LS BYTE
X B = SIGN/*D*, IN MS WORD
X SAVED THE OLD PAR IN MS WORD OF A3
X ALGEBRAICALLY ADD (P) + D
X CLEAR OLD PAR
X CREATE NEW PAR
X
X THIS ROUTINE GETS THE CONTENTS OF RCH
X THAT INDICATED IN PAR AND RETURNS
X IN RCH
X REFERENCE PAR2
X SAVE RETURN ADDRESS
X SELECT REGISTER STACK TO BE USED
X ADDR OF GEN REG STACK RETURNED IN PAR2
X GET CONTENTS OF RCH
X RESTORE RETURN ADDRESS
X PAR IS USED AS TEMP STORAGE
X FOR TEST PURPOSES
X
X C1950C00
X C1951000
X C1952000
X C1953000
X C1954000
X C1955000
X C1956000
X C1957000
X C1958000
X C1959000
X C196000
X C1961000
X C1962000
X C1963000
X C1964000
X C1965000
X C1966000
X C1967000
X C1968000
X C1969000
X C197000
X C1971000
X C1972000
X C1973000
X C1974000
X C1975000
X C1976000
X C1977000

X THIS ROUTINE ANALYZES THE *M* FIELD OF
X A TYPE RX INSTRUCTION. UPON ENTRANCE
X A3 CONTAINS THE MOST SIGNIF. WORD OF
X THE DOUBLEWORD INSTRUCTION. LC2
X INDICATES BYTE INSTRUCTION. UPON EXIT
X OF DAWI, DAWCI, OR INOINSTRUCTION
X R CONTAINS YR LC2 PASSES MS(C)*LS(1)
X PREPARE PS 2 BYTES OF A3 TO HOLD RETURN
X ADDRESS UPON ADDR IN A3(UPPER HALF)
X RESTORE A3 RETURN ADDR/INSTRUCTION
X MASK OFF *M* FIELD
X IS M = 0?
IF TRUE THEN STEP ELSE SKIP
DAWCI - 1 = MPCR
B GE0 LIT
8 = LIT
IF FALSE THEN STEP ELSE SKIP
DAWCI - 1 = MPCR
B = A2
IF NOT LST THEN STEP ELSE SKIP
INDIR - 1 = MPCR
X IF M = 10,12,14,16 THEN INOINSTRUCTION

```

R111

RH1

RXMF10D:


```

05A7 4809 18C1 A030 00F0
05A8 9000 0000 0030 002C
05A9 4809 CC56 8830 00F0
05AA 2000 0000 0070 0010
05AB 08AC 00C0 0030 0040

SRM14:
05AC 4809 18C1 A030 00F0
05AD 3000 0000 0030 002C
05AE 4809 CC56 8830 00F0
05AF 8000 0000 0070 0010
05B0 08AC 00C0 0030 0040

SRM16:
05B1 4809 18C1 A030 00F0
05B2 1C00 00C0 0030 0020
05B3 4809 CC56 8830 00F0
05B4 A000 0000 0070 0010
05B5 08AC 00C0 0030 0040

SRM18:
05B6 4809 CC5E 0030 00F0
05B7 3010 00C0 0030 0020
05B8 7419 2C52 0040 00F0
05B9 386C 0000 0030 0040
05BA 4820 0000 0030 0020
05BB 4820 0000 0030 0020
05BC 491C 0000 0030 0040
05BD 4100 00C0 0030 0040

05BE 4809 0003 0030 24F0
05BF 4809 0C40 0030 00F0
05C0 4809 0640 0800 00F0
05C1 4809 2003 0C1C 00F0
05C2 026C 0000 0030 0060
05C3 2F5C 00C0 0070 0060
05C4 4809 0F45 001C 00F0
05C5 4809 A0C0 0030 00F0
05C6 2F50 00C0 0030 0060
05C7 4809 0F45 001C 00F0
05C8 4809 0000 0030 00F0
05C9 2F50 00C0 0030 0060
05CA 4809 0F45 001C 00F0
05CB 4809 5000 0030 00F0
05CC 2F50 00C0 0030 0060
05CD 4809 0C40 0030 00F0
05CE 4809 0003 0030 00F0

* FROM SR02 AND CALL THE APPROPRIATE
* ISOLATE IM: FIELD = 12
* B CONTAINS STATUS REG 2
* P CONTAINS CONTENTS OF IM: FIELD
* JUMP TO TEST CONTENTS ROUTINE
* THIS ROUTINE WILL REMOVE THE IM: FIELD
* FROM SR02 AND CALL THE APPROPRIATE
* INDIRECT ADDRESSING ROUTINE
* ISOLATE IM: FIELD = 14
* B CONTAINS STATUS REG 2
* P CONTAINS CONTENTS OF IM: FIELD
* JUMP TO TEST CONTENTS ROUTINE
* THIS ROUTINE WILL REMOVE THE IM: FIELD
* FROM SR02 AND CALL THE APPROPRIATE
* INDIRECT ADDRESSING ROUTINE
* ISOLATE IM: FIELD = 16
* B CONTAINS STATUS REG 2
* P CONTAINS CONTENTS OF IM: FIELD
* JUMP TO TEST CONTENTS ROUTINE
* THIS ROUTINE ANALYSES THE IM: FIELD
* CONTENTS M = 0.142,3
* DIRECT ADDRESSING WITH INDEXING
* TEST IF "M" = 2
* JUMP TO INDIRECT ADDR WITH INDEXING
* INDIRECT ADDR WITHOUT INDEXING
* *****END RX FORMAT ROUTINES*****
* THIS ROUTINE SAVES LOGIC UNIT) REG.
* REFERENCE R02/HAR
* STORE B IN HAR
* SAVE RETURN IN B
* WRITE B INTO WORKSPACE
* WRITE A1 INTO WORKSPACE +1
* WRITE A2 INTO WORKSPACE + 2
* WRITE A3 INTO WORKSPACE + 3
* RESTORE ADDRESS
* STEP

```


[illegible]

[illegible]


```

0650 4809 00C1 0000 0000 0 02278C00 0
0651 4809 0000 0000 0000 0 02279C00 0
0652 4809 2045 0000 0000 0 02280C00 0
0653 4809 00C3 0000 0000 0 02281C00 0
0654 4809 00C3 0000 0000 0 02282C00 0
0655 4809 0F42 0035 0000 0 02283C00 0
0656 00C0 00C0 0030 0000 0 02284C00 0
0657 4809 0F43 801C 0000 0 02285C00 0
0658 0000 00C0 0030 0020 0 02286C00 0
0659 4809 00C0 0030 0000 0 02287C00 0
065A 4809 0F40 2040 0000 0 02288C00 0
065B 4809 0C43 001C 0000 0 02289C00 0
065C 4809 C000 0000 0000 0 02290C00 0
065D 4809 00C0 0000 0000 0 02291000 0
065E 4809 0040 2030 0000 0 02292C00 0
065F 4809 18C3 8830 0000 0 02293C00 0
0660 00C0 00C0 0030 0020 0 02294C00 0
0661 20C8 8000 0030 0000 0 02295C00 0
0662 4809 C05C 203C 0000 0 02296C00 0
0663 4809 C0C0 0030 0000 0 02297000 0
0664 4809 0000 0030 1000 0 02298C00 0
0665 4809 0155 2040 0000 0 02299C00 0
0666 00C0 00C3 0030 0000 0 02300C00 0
0667 00C0 00C3 0030 0000 0 02301000 0
0668 4809 C043 0030 0000 0 02302C00 0
0669 4809 C05C 201C 0000 0 02303C00 0
066A 4809 00C0 0030 0000 0 02304C00 0
066B 00C0 C000 0030 0000 0 02305C00 0
066C 20C8 C000 0030 0000 0 02306C00 0
066D 5260 00C3 0030 0000 0 02307C00 0
066E 5130 C000 0000 0000 0 02308C00 0

% GENERATE 512 IN B
% B NOW CONTAINS 768(ADDR OF PAGEREG)
% REFERENCE BR2
% SAVE HARTOFEET IN COUNTER
% PUT BR2(PAGE ADDR REG NO.) IN MAR2
% SAVE A2 IN MIR
% USE A2 AS X-SELECT INCLDF FOR BRAR
% MAR2 CONTAINS THE BASE ADDRESS OF THE
% DESIRED PAGE ADDRESS REGISTER
% READ CONTENTS OF PAGE ADDR. REG.
% READ COMPLETE STORE IN E
% SAVE PAGE ADDRESS REG CONTENTS IN A2
% SET PAGE MODIFICATION BIT (BIT 15)
STEP ELSE SKIP % STORE TO MEMORY, SET
% PAGE REFERENCED BIT IN PAGE ADDR REG
% FORM PAGE ADDRESS REG PLUS MOD. BIT
% SAVE RETURN IN R
% MODIFIED PAGE ADDR. REG. IN MIR
% WRITE OUT MODIFIED PAGE REG
% ISOLATE PAGE NUMBER
% MASK OFF LOW 8 BITS
% RESTORE A2
% R HAS THE PAGE REG. CONTENTS
% BR2 CONTAINS DIRECT ADDRESS (PAGEINC
% COMPLETE)
% SET IN COUNTER
% RETURN TO EMLIN
%
%
% ***** OFCODE IMPLEMENTATION *****
%
% ANALYZE THE INSTRUCTION
% THIS ROUTINE WILL FETCH AN INSTRUCTION
% AND(2317C00 0)
% SELECT THE OP(UPPER 6 BITS)
% SELECT THE CORRESPONDING ROUTINE FOR
% THE SELECTED OPERATION.
% A1 TO THE ADDR
% A1 TO THE ADDR
% EXAMINE REPOTE EXECUTE BIT
% GET THE NEXT INSTRUCTION
% CLEAR CONDITION BITS
% SAVE INSTRUCTION INTO A3
% PLACE THE 6 OP(UPPER 6)
% BITS OF A2
% CREATE ADDRESS OF OPERATION ROUTINE
% BASE ADDRESS OF OP(UPPER 6)
% SETTING OFCODE VALUE TO OFTABLE BASE
% JUMP TO AMPCR + 1
%
%
% THE FOLLOWING 64 INSTRUCTIONS CORRESPOND TO SUPROUTINE CALLS TO THE

```



```

040C 1080 0000 0000 0040
040D 1090 0000 0000 0040
05012 - 1 = MPCR
05013 - 1 = MPCR

0F010:
060E 4809 2C55 0000 00FC
060F 00F0 0000 0000 00FC
0610 51CC 00C3 0000 0060
0611 2F30 0000 0000 0060
0612 4809 00C0 0000 0060
0613 200C 0000 00C3 0060
0614 4809 00C0 0000 0060
0615 80FC 00C3 0000 0080
0616 4809 00C0 0000 0080
0617 51CC 00C3 0000 0060
0618 2F50 0000 0000 0060
0619 56E0 00C0 0000 0060

0F011:
06EA 4809 2C55 0030 00FC
06EB 00F0 0000 0000 00FC
06EC 51CC 00C3 0000 0060
06ED 2F30 0000 0000 0060
06EE 4809 00C4 0010 00FC
06EF 00C0 0000 0000 0030
06F0 60E0 0000 0000 0060
06F1 4809 00C0 0000 0060
06F2 200C 0000 0000 0060
06F3 4809 00C0 0000 0060
06F4 80F0 00C0 0000 0080
06F5 4809 2C55 0000 00FC
06F6 51CC 00C3 0000 0060
06F7 2F50 0000 0000 0060
06F8 56E0 00C0 0000 0060

0F012:
06F9 6330 00C0 0030 0060
06FA 4809 00C1 0000 00FC
06FB 00F0 0000 0000 00FC
06FC 4809 00C5 0000 0060
06FD 4809 00C0 0000 0060
06FE 4809 00C0 0000 0060
06FF 4809 00C5 0000 0060
0700 4809 00C0 0000 0060
0701 6000 0000 0000 0060
0702 56F0 0000 0000 0060
0703 4809 00C1 2000 0060
0704 0000 0000 0000 0060
0705 4809 0000 0000 0060
0706 4809 00C0 0000 0060
0707 200C 0000 0000 0060

0545800C 0
05458000 0
05458000 0
05461000 0
05461000 0
05462000 0
05463000 0
05464000 0
05465000 0
05466000 0
05467000 0
05468000 0
05469000 0
05470000 0
05471000 0
05472000 0
05473000 0
05474000 0
05475000 0
05476000 0
05477000 0
05478000 0
05479000 0
05480000 0
05481000 0
05482000 0
05483000 0
05484000 0
05485000 0
05486000 0
05487000 0
05488000 0
05489000 0
05490000 0
05491000 0
05492000 0
05493000 0
05494000 0
05495000 0
05496000 0
05497000 0
05498000 0
05499000 0
05500000 0
05501000 0
05502000 0
05503000 0
05504000 0
05505000 0
05506000 0
05507000 0
05508000 0
05509000 0
05510000 0
05511000 0
05512000 0
05513000 0
05514000 0
05515000 0
05516000 0
05517000 0

% RR TYPE INSTRUCTION
% CONTENTS OF R(H) STORED INTO R(A)
% SELECT LS 4 BITS OF B (*#* FIELD)
% SELECT REGISTER STACK TO BE USED
% ADDR OF GEN REG STACK RETURNED IN MAR2
% PUT CONTENTS OF R(H) IN B
% CONTENTS OF R(H) IN MIR
% ARITHMETIC CC SETTING (CENTER WITH B)
% OBTAIN *A* FIELD
% SELECT REGISTER STACK TO BE USED
% ADDR OF GEN REG STACK RETURNED IN PAR2
% CONTENTS OF R(H) INTO R(A)
% GET TYPE IT, (Y*) INTO R(A)
% OBTAIN INE *#* FIELD
% SELECT REGISTER STACK TO BE USED
% ADDR OF GEN REG STACK RETURNED IN MAR2
% R(CHH) = Y* PUT INTO D
% SET UP ADDRESS OF Y* IN ER2
% READ (Y*) INTO B
% (Y*) IN MIR
% ARITHMETIC CC SETTING (CENTER WITH B)
% EXTRACT *A* FIELD
% SELECT REGISTER STACK TO BE USED
% ADDR OF GEN REG STACK RETURNED IN MAR2
% WRITE OUT CONTENTS OF Y* INTO R(A)
% GET TYPE INSTRUCTION
% (Y) INTO R(A) IF B#
% (Y) (RCHH) = R(A) IF B# 0
% GET CONTENTS OF Y* FIELD
% PREPARE A3 FOR STORAGE OF Y* FIELD
% CLEAR MIR, BR2
% PLACE Y* FIELD IN LS 4 BITS OF A2
% PUT Y* FIELD INTO MAR
% CHECK IF *#* FIELD 0
% ANALYZE R(H) AND RETURN VALUE IN MIR
% ISOLATE Y INTO A2
% PUT Y INTO A2 AND (R(H)) INTO B
% B, MIR CONTAINS FUTURE CONTENTS OF R(A)
% ARITHMETIC CC SETTING (CENTER WITH B)

```


OP	OPCODE	OPNAME	DESCRIPTION
0706	4809 0000 0800 0000	43 R = B	% PUT 1A1 FIELD INTO L5 4 BITS OF B
0709	0800 0000 0000 0000	20 = SARF 15 = LIT	% PUT ADDR OF R(A) INTO B
0710	0800 2555 0800 0000	LIT AND B = 0	% SELECT REGISTER STACK TO BE USED
0711	4809 0000 0000 0000	REGSTACK - 1 = CPCR	% ADDR OF GEN REG STACK RETURNED IN MAR2
0712	5100 0000 0000 0000	REGSTACK - 1 = CPCR	% (RCH) + (Y) INTO R(A)
0713	0800 0000 0000 0000	REGSTACK - 1 = CPCR	% GET NEXT INSTRUCTION
0714	2F50 0000 0000 0000	OPCODE - 1 = CPCR	%
0715	6600 0000 0000 0000	OPCODE - 1 = MPCR	%
0716	5700 0000 0000 0000	OPCODE - 1 = CPCR	%
0717	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0718	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0719	6600 0000 0000 0000	OPCODE - 1 = MPCR	%
0720	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0721	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0722	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0723	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0724	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0725	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0726	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0727	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0728	2F50 0000 0000 0000	OPCODE - 1 = CPCR	%
0729	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0730	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0731	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0732	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0733	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0734	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0735	1100 0000 0000 0000	OPCODE - 1 = MPCR	%
0736	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0737	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0738	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0739	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0740	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0741	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0742	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0743	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0744	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0745	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0746	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0747	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0748	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0749	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0750	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0751	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0752	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0753	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0754	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0755	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0756	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0757	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0758	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0759	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0760	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0761	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0762	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0763	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0764	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0765	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0766	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0767	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0768	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0769	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0770	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0771	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0772	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
0773	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0774	5000 0000 0000 0000	OPCODE - 1 = MPCR	%
0775	1000 0000 0000 0000	OPCODE - 1 = MPCR	%
0776	4809 0000 0000 0000	OPCODE - 1 = CPCR	%
0777	0800 0000 0000 0000	OPCODE - 1 = CPCR	%
07			


```

07C6 4809 2001 0820 00F0
07C7 4809 CC40 0C30 00F0
07C8 78C9 00C0 0090 00F0
07C9 1E7C 00C0 0000 0060
07CA 4F1C 00C0 0070 0060
07CB 4809 00C0 0090 00F0
07CC 4809 0C40 8B30 00F0
07CD 00C0 00C0 0090 0020
07CE 200C 00C0 0070 0060
07CF 2F5F 00C0 0000 0060
07D0 66EC 00C0 0030 0040

```

0P02013:

```

07D1 4809 0C40 2C4C 00F0
07D2 0020 00C0 00C0 0040
07D3 4809 2001 0820 00F0
07D4 4809 CC5E 0030 00F0
07D5 78C9 00C0 0000 0060
07D6 1E7C 00C0 00C0 0060
07D7 4F1C 00C0 0070 0060
07D8 48C9 00C0 0090 00F0
07D9 4809 0C40 8B30 00F0
07DA 00C0 00C0 0090 0020
07DB 200C 00C0 0070 0060
07DC 2F5F 00C0 0000 0060
07DD 66EC 00C0 0030 0040

```

0P021:

```

07DE 4809 2C55 0830 00F0
07DF 00F0 00C0 00C0 00E0
07E0 51C0 00C0 0050 0060
07E1 2F3C 00C0 00C0 0060
07E2 48C0 00C0 0070 0060
07E3 66EC 00C0 0030 0040

```

0P023:

```

07E4 520C 00C0 0030 0060
07E5 4820 00C0 0020 0060
07E6 56E0 00C0 0C70 004C

```

0P0CODE03:

```

07E7 5F9C 00C0 00C0 0060
07E8 4809 CC40 0040 00FC
07E9 118C 00C0 0030 00CC
07EA 48C9 00C0 0000 00F0
07EB 4824 00C0 0030 00F0

```

0P03F:

```

07EC 11C0 07C0 00C0 0040
07ED 300C 00C0 0C7C 0040
07EE 300C 07C0 00C0 0040
07EF 11D0 C0C0 0C70 0040

```

0P030:

```

07F0 4809 2C56 2030 00F0
07F1 00FC 00C0 0030 00E0
07F2 48C9 C640 0010 00FC
07F3 11E0 00C0 0070 00CC

```

```

LIT L = 0
A2 + B = MIR
% CHECK FOR CARRY
02755C00 0
02760E00 0
02761C00 0
02762C00 0
02763C00 0
02764C00 0
02765C00 0
02766C00 0
02767C00 0
02768C00 0
02769C00 0
02770C00 0
02771C00 0
02772C00 0
02773C00 0
02774C00 0
02775C00 0
02776C00 0
02777C00 0
02778C00 0
02779C00 0
02780C00 0
02781C00 0
02782C00 0
02783C00 0
02784C00 0
02785C00 0
02786C00 0
02787C00 0
02788C00 0
02789C00 0
02790C00 0
02791C00 0
02792C00 0
02793C00 0
02794C00 0
02795C00 0
02796C00 0
02797C00 0
02798C00 0
02799C00 0
02800C00 0
02801C00 0
02802C00 0
02803C00 0
02804C00 0
02805C00 0
02806C00 0
02807C00 0
02808C00 0
02809C00 0
02810C00 0
02811C00 0
02812C00 0
02813C00 0
02814C00 0
02815C00 0
02816C00 0
02817C00 0

% DECREASE RA BY TWO, TYPE RR
% (RA) - 2 -> RA
% B = (RA); MAR2 = RA
02771C00 0
02772C00 0
02773C00 0
02774C00 0
02775C00 0
02776C00 0
02777C00 0
02778C00 0
02779C00 0
02780C00 0
02781C00 0
02782C00 0
02783C00 0
02784C00 0
02785C00 0
02786C00 0
02787C00 0
02788C00 0
02789C00 0
02790C00 0
02791C00 0
02792C00 0
02793C00 0
02794C00 0
02795C00 0
02796C00 0
02797C00 0
02798C00 0
02799C00 0
02800C00 0
02801C00 0
02802C00 0
02803C00 0
02804C00 0
02805C00 0
02806C00 0
02807C00 0
02808C00 0
02809C00 0
02810C00 0
02811C00 0
02812C00 0
02813C00 0
02814C00 0
02815C00 0
02816C00 0
02817C00 0

% LOAD DOUBLE, TYPE R(CLI)
% (Y*Y* + 1) -> RA;RA+1; SET CC
% ISOLATE IN: FIELD
02786C00 0
02787C00 0
02788C00 0
02789C00 0
02790C00 0
02791C00 0
02792C00 0
02793C00 0
02794C00 0
02795C00 0
02796C00 0
02797C00 0
02798C00 0
02799C00 0
02800C00 0
02801C00 0
02802C00 0
02803C00 0
02804C00 0
02805C00 0
02806C00 0
02807C00 0
02808C00 0
02809C00 0
02810C00 0
02811C00 0
02812C00 0
02813C00 0
02814C00 0
02815C00 0
02816C00 0
02817C00 0

% RESULT IN LHM OF B AND MIR
02781C00 0
02782C00 0
02783C00 0
02784C00 0
02785C00 0
02786C00 0
02787C00 0
02788C00 0
02789C00 0
02790C00 0
02791C00 0
02792C00 0
02793C00 0
02794C00 0
02795C00 0
02796C00 0
02797C00 0
02798C00 0
02799C00 0
02800C00 0
02801C00 0
02802C00 0
02803C00 0
02804C00 0
02805C00 0
02806C00 0
02807C00 0
02808C00 0
02809C00 0
02810C00 0
02811C00 0
02812C00 0
02813C00 0
02814C00 0
02815C00 0
02816C00 0
02817C00 0

% SET/CLEAR CARRY BIT
% CHECK FOR OVERFLOW
02778C00 0
02779C00 0
02780C00 0
02781C00 0
02782C00 0
02783C00 0
02784C00 0
02785C00 0
02786C00 0
02787C00 0
02788C00 0
02789C00 0
02790C00 0
02791C00 0
02792C00 0
02793C00 0
02794C00 0
02795C00 0
02796C00 0
02797C00 0
02798C00 0
02799C00 0
02800C00 0
02801C00 0
02802C00 0
02803C00 0
02804C00 0
02805C00 0
02806C00 0
02807C00 0
02808C00 0
02809C00 0
02810C00 0
02811C00 0
02812C00 0
02813C00 0
02814C00 0
02815C00 0
02816C00 0
02817C00 0

% RESULT IN LHM OF B AND MIR
02781C00 0
02782C00 0
02783C00 0
02784C00 0
02785C00 0
02786C00 0
02787C00 0
02788C00 0
02789C00 0
02790C00 0
02791C00 0
02792C00 0
02793C00 0
02794C00 0
02795C00 0
02796C00 0
02797C00 0
02798C00 0
02799C00 0
02800C00 0
02801C00 0
02802C00 0
02803C00 0
02804C00 0
02805C00 0
02806C00 0
02807C00 0
02808C00 0
02809C00 0
02810C00 0
02811C00 0
02812C00 0
02813C00 0
02814C00 0
02815C00 0
02816C00 0
02817C00 0

% LOAD DOUBLE, TYPE RX
% (Y*Y* + 1) -> RA;RA+1
% B = Y
% CALL LOAD DOUBLE ROUTINE
02795C00 0
02796C00 0
02797C00 0
02798C00 0
02799C00 0
02800C00 0
02801C00 0
02802C00 0
02803C00 0
02804C00 0
02805C00 0
02806C00 0
02807C00 0
02808C00 0
02809C00 0
02810C00 0
02811C00 0
02812C00 0
02813C00 0
02814C00 0
02815C00 0
02816C00 0
02817C00 0

% THIS ROUTINE ANALYZES THE 03 OP CODE
%
02808C00 0
02809C00 0
02810C00 0
02811C00 0
02812C00 0
02813C00 0
02814C00 0
02815C00 0
02816C00 0
02817C00 0

B AND LIT = A2
LIT = LIT
A2 + AMPC = AMPCR
0P030H - 1 = AMPCR

```


168

0023	4809	£000	9000	0010
0024	90F0	0000	0000	008C
0025	4809	£156	0030	00F0
0026	51C0	0000	0030	0060
0027	2F30	0000	0030	0060
0028	4809	A000	C030	00CF
0029	0000	00C0	0030	0020
002A	4809	A001	4000	00F0
002B	4809	AC50	4030	00F0
002C	56E0	00C0	0000	0040

[illegible]

08943	340F	E00D	9700	C6FC
08944	30FD	0000	0000	00B0
08945	480F	E156	0830	00F0
08946	31CC	00C0	0C70	06C0
08947	2F30	00C0	0C70	06AC
08948	480F	0C40	2E30	00F0
08949	480F	20D3	0C1F	00F0
08950	022C	0000	093C	00C0
08951	2F3C	0000	0000	0060
08952	48F9	0C40	8B30	00F0
08953	00C0	0000	0000	0020
08954	480F	0C41	0000	00F0
08955	480F	C5C5	C030	00F0
08956	22C0	0000	0000	0060
08957	5AED	0000	0030	00D0
08958	S130	0CC3	00C0	0060
08959	480F	0C41	087C	00FF
08960	00F0	0000	0000	00A0
08961	480F	E5C5	1000	00F0

```

A3 M = A3
q = SARI 15 = LII
A3 AND LII = B
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
A1 R = A1
16 = SAR
A1 L = A1
A1 OR B = A1
OPCODE - 1 = MPCR

```

```

0F03005:1  % LOAD SR1 % (R(A)) INTO SF1

A3 R = A3
4 = SARP 15 = LIT
A3 AND LIT = B
REGSTACK - 1 = CPCR
INPUT - 1 = CPCR
B L = MIR
COMP 16 = SAR
LIT L = A2
7 = LIT/ COMP ?9 = SAR
LIT L = B
112 = LIT/ COMP 16 = SAR
A2 OR B = B
NOT B = A2, BHI
A2 AND B = MIR
NOT A2 = B
A1 AND B = A2, BHI
A2 OR B = B
A2 = A1
COMP 16 = SAR
A1 B = A1
A1 OR B = A1
REGCODE - 1 = MPCR
% RESTORE A1
% RESTORE A1 (ACTIVE PSW)
% UNDOUCHED
% CLEAR UHW OF A1
% CREATE NEW PART OF A1 INTO A2
% MASK OFF SAVED PART OF A1 INTO A2
% MASK FOR A1 (PSW)
% MASK OFF (RA), STORE IN MIR
% MASK FOR (RA)
% CREATE MASK ONES IN UHW GF B
% LOWER MASK OF SR1
% CREATE BIT MASK FOR LOADER TOGGLES
% STORE (R(A)) IN UHW OF MIR
% B = (R(A))
% RA IN MAR2
% LOAD SR1 % (R(A)) INTO SF1

```

```

OP0306G:
A3 R = A3
A3 = SARJ 15 = LIT
A3 AND LIT = B
REGSTACK - 1 = CPCR
INPUT - 1 = CPCR
B = A2
LIT = MAR2
LIT - 32 = LIT
INPUT - 1 = CPCR
B R = B
16 = SAR
B L = B
A2 OR B = MIR
EINPUT - 1 = CPCR
OPCODE - 1 = MPCR

OP033:
IFETCH - 1 = CPCR
B L = B
COMP 16 = SARJ 15 = LIT
A3 OR B = A3

% LOAD SR2, (R(A)) INTO SR2
% PA = MAR2
% B = (R(A))
% (R(A)) = A2
% R = (STATUS2)
% ISOLATE UPPER 16 BITS OF STATUS2
% NEW STATUS2 CREATED
% (R(A)) INTO STATUS2
%
%
% LOAD MULTIPLE, TYPE RX
% Y ... Y = H - A + 1 INTO RA ... RH
% B = ITI = Y
%
% STORE Y IN UNIV OF A3

```

0.23795000	0.23805000	0.23815000	0.23825000	0.23835000	0.23845000	0.23855000	0.23865000	0.23875000	0.23885000	0.23895000	0.23905000	0.23915000	0.23925000	0.23935000	0.23945000	0.23955000	0.23965000	0.23975000	0.23985000	0.23995000	0.24005000	0.24015000	0.24025000	0.24035000	0.24045000	0.24055000	0.24065000	0.24075000	0.24085000	0.24095000	0.24105000	0.24115000	0.24125000	0.24135000	0.24145000	0.24155000	0.24165000	0.24175000	0.24185000	0.24195000	0.24205000	0.24215000	0.24225000	0.24235000	0.24245000	0.24255000	0.24265000	0.24275000	0.24285000	0.24295000	0.24305000	0.24315000	0.24325000	0.24335000	0.24345000	0.24355000	0.24365000	0.24375000	0.24385000	0.24395000	0.24405000	0.24415000	0.24425000	0.24435000	0.24445000	0.24455000	0.24465000	0.24475000	0.24485000	0.24495000	0.24505000	0.24515000	0.24525000	0.24535000	0.24545000	0.24555000	0.24565000	0.24575000	0.24585000	0.24595000	0.24605000	0.24615000	0.24625000	0.24635000	0.24645000	0.24655000	0.24665000	0.24675000	0.24685000	0.24695000	0.24705000	0.24715000	0.24725000	0.24735000	0.24745000	0.24755000	0.24765000	0.24775000	0.24785000	0.24795000	0.24805000	0.24815000	0.24825000	0.24835000	0.24845000	0.24855000	0.24865000	0.24875000	0.24885000	0.24895000	0.24905000	0.24915000	0.24925000	0.24935000	0.24945000	0.24955000	0.24965000	0.24975000	0.24985000	0.24995000	0.25005000	0.25015000	0.25025000	0.25035000	0.25045000	0.25055000	0.25065000	0.25075000	0.25085000	0.25095000	0.25105000	0.25115000	0.25125000	0.25135000	0.25145000	0.25155000	0.25165000	0.25175000	0.25185000	0.25195000	0.25205000	0.25215000	0.25225000	0.25235000	0.25245000	0.25255000	0.25265000	0.25275000	0.25285000	0.25295000	0.25305000	0.25315000	0.25325000	0.25335000	0.25345000	0.25355000	0.25365000	0.25375000	0.25385000	0.25395000	0.25405000	0.25415000	0.25425000	0.25435000	0.25445000	0.25455000	0.25465000	0.25475000	0.25485000	0.25495000	0.25505000	0.25515000	0.25525000	0.25535000	0.25545000	0.25555000	0.25565000	0.25575000	0.25585000	0.25595000	0.25605000	0.25615000	0.25625000	0.25635000	0.25645000	0.25655000	0.25665000	0.25675000	0.25685000	0.25695000	0.25705000	0.25715000	0.25725000	0.25735000	0.25745000	0.25755000	0.25765000	0.25775000	0.25785000	0.25795000	0.25805000	0.25815000	0.25825000	0.25835000	0.25845000	0.25855000	0.25865000	0.25875000	0.25885000	0.25895000	0.25905000	0.25915000	0.25925000	0.25935000	0.25945000	0.25955000	0.25965000	0.25975000	0.25985000	0.25995000	0.26005000	0.26015000	0.26025000	0.26035000	0.26045000	0.26055000	0.26065000	0.26075000	0.26085000	0.26095000	0.26105000	0.26115000	0.26125000	0.26135000	0.26145000	0.26155000	0.26165000	0.26175000	0.26185000	0.26195000	0.26205000	0.26215000	0.26225000	0.26235000	0.26245000	0.26255000	0.26265000	0.26275000	0.26285000	0.26295000	0.26305000	0.26315000	0.26325000	0.26335000	0.26345000	0.26355000	0.26365000	0.26375000	0.26385000	0.26395000	0.26405000	0.26415000	0.26425000	0.26435000	0.26445000	0.26455000	0.26465000	0.26475000	0.26485000	0.26495000	0.26505000	0.265
------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	-------


```

C856 4809 ELS6 2000 00F0      02938100 0
C857 4809 ERO3 8800 00F0      02939100 0
C858 3000 00C0 00C0 0000      02940100 0
C859 4809 2556 053A 00F0      02941000 0
C85A 4809 C55E 0000 CFCF      02942000 0
C85B 0100 00C0 00C0 00E0      02943000 0
C85C 7000 0000 0000 00F0      02944000 0
C85D 4809 C45E 2000 00F0      02945000 0
C85E 4809 C55E 2000 00F0      02946000 0
C85F 4809 C012 00C0 00F0      02947000 0
C860 4809 C012 00C0 00F0      02948000 0
C861 4809 C0C0 20C0 20F0      02949000 0
C862 51C0 0000 0000 0060      02950000 0
C863 4809 0F41 00C0 40F0      02951000 0
C864 0000 0000 00C0 0030      02952000 0
C865 4809 E003 9010 C0FF      02953000 0
C866 4809 E0C0 9020 00F0      02954000 0
C867 60C0 00C0 00C0 0060      02955000 0
C868 4809 0C43 00C0 C0F0      02956000 0
C869 4809 0C43 00C0 20F0      02957000 0
C86A 4809 0F43 581C C0F0      02958000 0
C86B 0000 00C0 00C0 0010      02959000 0
C86C 2F50 00C0 0000 0060      02960000 0
C86D 4809 0C46 0800 C0F0      02961000 0
C86E 51C0 00C0 0000 C060      02962000 0
C86F 4809 0F41 00C0 00F0      02963000 0
C870 0000 0000 00C0 0030      02964000 0
C871 4809 E0C0 10C0 C0F0      02965000 0
C872 4809 E001 0010 00F0      02966000 0
C873 4809 C00E 2000 00FC      02967000 0
C874 4809 C012 00C0 00F0      02968000 0
C875 5810 00C0 0000 00F0      02969000 0
C876 4809 C0C0 00C0 00F0      02970000 0
C877 56EC 00C0 00C0 0040      02971000 0
C878 5F90 0000 00C0 0060      02972000 0
C879 4809 C640 00C0 C0F0      02973000 0
C87A 1250 0000 00C0 00C0      02974000 0
C87B 4809 C0C3 00C0 00F0      02975000 0
C87C 4824 00C0 C0C0 00F0      02976000 0
C87D 1260 00C0 00C0 C040      02977000 0
C87E 3000 0000 00C0 C040      02978000 0
C87F 3000 00C0 C000 C040      02979000 0
C880 1270 00C0 00C0 C040      02980000 0
C881 4809 2556 2000 00FC      02981000 0
C882 00FC 0000 0000 00EC      02982000 0
C883 4809 C45E 00C0 00F0      02983000 0
C884 00C0 00C0 C0C0 C0E0      02984000 0
C885 7800 00C0 00C0 00FC      02985000 0
C886 3000 0000 C0C0 C040      02986000 0
C887 4809 C640 0040 00F0      02987000 0
C888 4809 C0C0 00C0 00C0      02988000 0
C889 00C0 00C0 0000 00F0      02989000 0
C88A 4824 00C0 00C0 C0F0      02990000 0

```

```

A3 AND LIT = A2      % A2 = 1H1 FIELD
A3 R = B
4 = SAR
LIT AND B = B
A2 LSS B
16 = LIT
IF TRUE THEN STEP ELSE SKIP
A2 + LIT = A2
A2 R = A2
12 COL0 THEN SKIP
A2 + 1 = A21 ASE
REGSTACK - 1 = CPCR
BMAR L = BRL+CSAR
CMP B = SAR
A3 R = A3+BR2
EMULIN - 1 = CPCR
B = MIR
BMAR R = B+MAR2
EDUTPUT - 1 = CPCR
B + 1 = B
REGSTACK - 1 = CPCR
BMAR L = BRL
CMP B = SAR
A3 + 1 = A3
A3 L = BR2
A2 - 1 = A2
A2 EOL 0
IF TRUE THEN SKIP
LIT AND B = B
DPCODE - 1 = MPCCR
DPCODE - 1 = MPCCR

OPCODE04:
XFCODE - 1 = CPCR
A2 + AMPCR = AMPCR
DP04F - 1 = AMPCR
STEP
EXEC
OP040:
OP040 - 1 = MPCCR
FAULT - 1 = MPCCR
FAULT - 1 = MPCCR
OP043 - 1 = MPCCR

OP040:
LIT AND B = A2
15 = LIT
A2 GE0 LIT
4 = LIT
IF TRUE THEN STEP ELSE SKIP
FAULT - 1 = MPCCR
A2 + AMPCR = AMPCR
DP04H - 1 = AMPCR
STEP
EXEC

```

```

02938100 0
02939100 0
02940100 0
02941000 0
02942000 0
02943000 0
02944000 0
02945000 0
02946000 0
02947000 0
02948000 0
02949000 0
02950000 0
02951000 0
02952000 0
02953000 0
02954000 0
02955000 0
02956000 0
02957000 0
02958000 0
02959000 0
02960000 0
02961000 0
02962000 0
02963000 0
02964000 0
02965000 0
02966000 0
02967000 0
02968000 0
02969000 0
02970000 0
02971000 0
02972000 0
02973000 0
02974000 0
02975000 0
02976000 0
02977000 0
02978000 0
02979000 0
02980000 0
02981000 0
02982000 0
02983000 0
02984000 0
02985000 0
02986000 0
02987000 0
02988000 0
02989000 0
02990000 0
02991000 0
02992000 0
02993000 0
02994000 0
02995000 0
02996000 0
02997000 0
02998000 0
02999000 0
0299A000 0
0299B000 0
0299C000 0
0299D000 0
0299E000 0
0299F000 0

```



```

052C 5F90 00C0 0000 0060
052D 4809 C440 0040 C0FC
052E 1324 00C0 00C0 00C0
052F 4809 00C0 0000 0060
0530 4824 00C0 007C C0FC

0531 133C 00C3 0000 0040
0532 1340 00C3 0000 0040
0533 3000 00C0 007C C04C
0534 133C 00C3 0000 0040

0535 4809 0C43 887C 00F0
0536 80F0 00C3 0000 0060
0537 4809 2C55 0830 C0F0
0538 51C0 0000 0070 C060
0539 2F30 0003 0C70 C060
053A 4809 0C40 2C00 00F0
053B 4809 E156 0800 00F0
053C 00F0 00C0 0000 00E0
053D 4809 0C5E 0000 80F0
053E 4809 00C1 0830 00F0
053F 4809 C55C 089C 00F0
0540 2F50 00C0 0070 C060
0541 200C 0003 0070 0060
0542 56E0 00C3 0000 0040

0543 4809 2C55 0830 00F0
0544 00F0 00C3 0000 00E0
0545 51C0 00C3 0070 0060
0546 2F30 00C0 0000 0060

0547 4805 0C41 CC10 C0F0
0548 00C0 00C0 0000 C030
0549 60C0 0003 0000 0060
054A 4809 CC40 0030 00F0
054B 1C80 00C0 0000 0060
054C 51C0 00C0 0000 0060
054D 2F50 00C0 0000 0060
054E 4809 00C0 C000 C0F0
054F 200C 0000 0000 0060
0550 308B C000 0000 C0FC
0551 56E0 0000 C030 C040
0552 4809 C000 C800 00F0
0553 51C0 0000 0000 0060
0554 2F3C 0000 0000 0060
0555 4809 CC45 0030 00F0
0556 2F50 0000 0000 0060
0557 56E0 00C0 0000 0040

0558 5700 00C3 0000 C060
0559 946C 00C3 0000 C040

OPCODE051
THIS ROUTINE ANALYZES THE 05 OPCODE
XFCODE - 1 = CPCR
A2 * AMPCR = AMPCR
OP05F - 1 = AMPCR
EXEC
OP05F1
OP05A - 1 = MPCR
OP05B - 1 = MPCR
FAULT - 1 = MPCR
OP053 - 1 = MPCR

OP0501
B R = B
4 = SARI 15 = LIT
LIT AND B = B
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
B = A2
A3 AND LIT = B
15 = LIT
-B = SAR
8001 L = B
A2 OR B = MIR4
EOUTPUT - 1 = CPCR
SETCCA - 1 = CPCR
OPCODE - 1 = MPCR

OP0511
LIT AND B = B
15 = LIT
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
B L = RR2
COMB 8 = SAR
EYULIN - 1 = CPCR
B = MIR
A2OM - 1 = CPCR
REGSTACK - 1 = CPCR
EOUTPUT - 1 = CPCR
BHI
SETCCA - 1 = CPCR
IF LC2 THEN STEP ELSE SKIP % LC2 SET IN A2OM
OPCODE - 1 = MPCR
A2 = B
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
R + 1 = MIR
EOUTPUT - 1 = CPCR
OPCODE - 1 = MPCR

LDINX11
B L = RR2
COMB 8 = SAR
EYULIN - 1 = CPCR
B = MIR
A2OM - 1 = CPCR
REGSTACK - 1 = CPCR
EOUTPUT - 1 = CPCR
BHI
SETCCA - 1 = CPCR
IF LC2 THEN STEP ELSE SKIP % LC2 SET IN A2OM
OPCODE - 1 = MPCR
A2 = B
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
R + 1 = MIR
EOUTPUT - 1 = CPCR
OPCODE - 1 = MPCR

OP0531
RYNFIELD - 1 = CPCR
E = Y
LDINX1 - 1 = MPCR

```



```

0988 139C 0009 0030 C04C
0989 13CC 0000 0000 C04D
098A 3000 00C0 0036 0040
098B 1300 00C0 0000 C040

0P07F1
0P070 - 1 = MPCR
0P071 - 1 = MPCR
FAULT - 1 = MPCR
0P073 - 1 = MPCR

0P0701
R R = B
R SARI 15 = LIT
LIT AND B = B
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
B = A2
A3 AND LIT = B
15 = LIT
R = SAR
A2 R = A2
IF LET THEN STEP ELSE SKIP X TEST BIT
TEST11 - 1 = MPCR
LIT EOL B
IF TRUE THEN B010 C = B SKIP X SET 00 INTO CC LITS
B011 C = B
B = SAR
A1 AND B = A1
OPCODE - 1 = MPCR

TEST11
LIT EOL B
IF TRUE THEN B010 C = B SKIP X SET 11 IN BOTH LC BITS
B010 C = B
A1 AND B = A1
OPCODE - 1 = MPCR

0P0711
LIT AND B = B
15 = LIT
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
B = A2
B L = BR2
COMP B = SAR
EYULIN - 1 = CPCR
A1 R = A1
16 = SAR
A1 L = A1
A1 OR B = A1
A2 + 1 = A2
A2 L = BR2
COMP B = SAR
EYULIN - 1 = CPCR
B L = MIN
COMP 16 = SAR
LIT L = A3
LIT B
LIT COMP 29 = SAR
LIT B
LIT COMP 16 = SAR
A3 OR B = B

09A5 4809 2C55 0800 00F0
09A6 00F0 00C9 0C70 00E0
09A7 51C0 00C0 0C00 00E0
09A8 2F3C 00C0 0C70 00E0
09A9 4809 00C0 20C0 00E0
09AA 48C9 0C41 001C 00E0
09AB 00C0 00C9 0C00 C030
09AC 50E0 00C0 0C00 00E0
09AD 4809 48C0 0C00 00E0
09AE 00C0 00C9 0C00 C020
09AF 48C9 40C1 4C00 00E0
09B0 4809 4C5C 4030 00E0
09B1 4809 00C0 2F00 00E0
09B2 4809 0C00 0C40 00E0
09B3 00C0 0C00 0C30 C030
09B4 50E0 00C0 0C30 C060
09B5 48C9 0C41 0C30 00E0
09B6 00C0 0C00 00C0 C020
09B7 48C9 00C0 0C00 00E0
09B8 307C 00C0 0C00 00E0
09B9 4809 20C1 4030 00E0
09BA 0700 00C0 0C30 C040
09BB 4809 4C5C 0800 00E0

```



```

00F8 6330 0000 0030 0060
00F9 4809 0643 0030 00F0
00EA 4809 E156 0830 00F0
00EB 00F0 0000 00C0 00F0
00EC 4809 0C52 00C0 00F0
00ED 6C19 0000 0800 00F0
00EE 238C 0000 0000 0060
00EF 4809 0C40 2000 00C0
00F0 4809 CC40 0C40 00F0
00F1 4809 EC52 0800 00F0
00F2 228C 00C0 0000 0060
00F3 4809 0C40 1030 00F0
00F4 4809 0C40 0030 00F0
00F5 4809 E0C0 4830 00F0
00F6 2F50 0000 00C0 0060
00F7 200C 0000 0000 0060
00F8 56E0 0000 0030 0040

00F9 4849 00C0 0030 0060
00FA 5700 00C0 0030 0060
00FB 4809 0C41 0C10 00F0
00FC 0000 00C0 00C0 0030
00FD 4809 0F40 0020 00F0
00FE 50EC 00C0 0000 0060
00FF 4809 0C40 0C30 00F0
0A00 4809 E0C0 0E7C 00F0
0A01 228C 0000 0020 0060
0A02 4809 2C57 100C 00F0
0A03 00FC 0000 0000 0060
0A04 380B 00C0 0000 00F0
0A05 143C 0000 0030 0040
0A06 4809 0C41 0000 00F0
0A07 0000 0000 00C0 0010
0A08 4809 0C40 4800 00F0
0A09 4809 E000 3000 00F0
0A0A 0000 00C0 0030 0010
0A0B 4809 EC52 1030 00F0
0A0C 4809 EC52 00C0 00F0
0A0D 1440 0000 0030 0040

0A0E 4809 0C40 4800 00F0
0A0F 0000 00C0 0030 0010
0A10 4809 0C41 0B20 00F0
0A11 4809 E0C0 3000 00F0
0A12 4809 EC5C 0C30 00F0
0A13 4809 0000 0020 00F0
0A14 4809 0F40 0010 00F0
0A15 51EC 00C0 0000 0060
0A16 56E0 00C0 00C0 0040

0A17 5F90 0000 00C0 0060
0A18 4809 C640 0030 00F0
0A19 145C 00C0 0030 00C0

00F103:
SEI LC2
RMFIELD - 1 = CPCR
B L = BR2
COMP 0 = SAR
EHAR = BR1
EHULIN - 1 = CPCR
B = MIR
A3 = B
CONTENISRA - 1 = CPCR
LIT AND B L = A3, BHI
255 = LIT, COMP 0 = SAR
IF LC2 THEN STEP ELSE SKIP
LS103 - 1 = MPCR
B L = B, CSAR
COMP 24 = SAR
B R = B
A3 R = A3, CSAR
0 = SAR
B L = B, CSAR
A3 = A3
A3 OR B = MIR
CNT103 - 1 = MPCR
B R = B, CSAR
B = SAR
B L = B, CSAR
A3 R = A3
A3 OR B = MIR
CNT103: ASR
EHAR = BR2
EHULOUT - 1 = MPCR
OPCODE - 1 = MPCR

LS1C3:
0A0E 4809 0C40 4800 00F0
0A0F 0000 00C0 0030 0010
0A10 4809 0C41 0B20 00F0
0A11 4809 EC5C 0C30 00F0
0A12 4809 E0C0 3000 00F0
0A13 4809 0000 0020 00F0
0A14 4809 0F40 0010 00F0
0A15 51EC 00C0 0000 0060
0A16 56E0 00C0 00C0 0040

00F00000:
X SHIFT (R(A3)) MUMI 1 BITS 0-5 PLAYS
X ZERO FILL AND SET CC
X Y INTO B
X ISOLATE *M= FIELD
X (R(A3)) INTO B
X TRANSFER (R(M)) INTO A2
X Y INTO MIR
X INSTRUCTION INTO B
X (R(A3)) INTO B
X (R(A3)) INTO A3
X (Y0-5) INTO SAR
X PERFORM SHIFT
X SET APPROPRIATE CONDITION BITS
X
X RX TYPE EYIE STORE
X (R(A3)) BITS 7 INTO Y BYTE
X LC2 USES A3 BYTE OPERATION FLAG
X Y ANDR INTO B
X Y INTO BR2
X STORE Y
X (Y) INTO B
X (Y) INTO MIR
X RESTORE INSTRUCTION INTO B
X (R(A3)) INTO B
X (R(A3)) BITS (0-7) INTO 2ND LSB OF A3
X (Y) INTO B
X IF LC2 SET, PUT BYTE INTO
X LS BYTE OF Y
X CLEAR THE MS BYTE OF B
X PUT (R(A3)) -BITS 0-7 INTO 2ND LS BYTE
X IN A3
X (R(A3)) INTO B
X MIR CONTAINS NEW (Y)
X USED AS A "GO TO"
X (R(A3)) BITS 0-7 INTO LS EYIE OF A3
X CLEAR LS BYTE OF B
X (R(A3)) BITS 0-7 INTO LS EYIE OF A3
X MIR CONTAINS NEW (Y)
X REFERENCE BR1
X Y INTO BR2
X WRITE INTO R(A3)
X
X
X RIGHT SHIFT AND STORE
X *F= INTO A2

```


0A14	4809	0000	0030	00F0					03478C00	0
0A10	4824	0000	0030	00F0					03479000	0
									03480000	0
									03481000	0
0A1C	146C	0000	0070	00H0					03482000	0
0A10	147C	0000	0030	00H0					03483000	0
0A1E	148C	0000	0070	00H0					03484000	0
0A1F	1490	0000	0030	00H0					03485000	0
									03486000	0
									03487000	0
									03488000	0
									03489000	0
									03490000	0
									03491000	0
									03492000	0
									03493000	0
									03494000	0
									03495000	0
									03496000	0
									03497000	0
									03498000	0
									03499000	0
									03500000	0
									03501000	0
									03502000	0
									03503000	0
									03504000	0
									03505000	0
									03506000	0
									03507000	0
									03508000	0
									03509000	0
									03510000	0
									03511000	0
									03512000	0
									03513000	0
									03514000	0
									03515000	0
									03516000	0
									03517000	0
									03518000	0
									03519000	0
									03520000	0
									03521000	0
									03522000	0
									03523000	0
									03524000	0
									03525000	0
									03526000	0
									03527000	0
									03528000	0
									03529000	0
									03530000	0
									03531000	0
									03532000	0
									03533000	0
									03534000	0
									03535000	0
									03536000	0
									03537000	0

OP1101:										
	CONTENTS - 1 = CPCR									
	B = MIR									
	A3 = B									
	CONTENTS - 1 = CPCR									
	B = A3, BHI									
	A3 C = A3, CSAR									
	15 = SAR									
	A3									
	IF LST THEN SET LCI									
	A3 C = A3									
	B = A2									
	ROT 0 = B									
	Z L = B									
	16 = SAR									
	16 LCI THEN A3 OR B = A3									
	A2 = SAR									
	A3 R = B									
	B L = B									
	COMP 16 = SAR									
	D R = B, MIR									
	EQUTPUT - 1 = CPCR									
	SETCCA - 1 = CPCR									
	OPCODE - 1 = MPCR									
OP111:										
	CONTENTS - 1 = CPCR									
	B = MIR									
	CONTENTS - 1 = CPCR									
	B L = B, R2									
	COMP 0 = SAR									
	EMULOUT - 1 = CPCR									
	OPCODE - 1 = MPCR									
OP112:										
	LIT AND B = 6									
	15 = LIT									
	B EOL 0									
	IF TRUE THEN SKIF									
	CONTENTS - 1 = CPCR									
	B L = B									
	COMP 16 = SAR									
	A3 OR 0 = A3									
	IFEITCH - 1 = CPCR									
0A3E	4809	2C56	0070	00F0						
0A3F	00F0	00C0	0070	00F0						
0A40	4809	0C52	0070	00F0						
0A41	6810	0000	0000	00F0						
0A42	2380	0000	0070	00F0						
0A43	4809	0C41	0070	00F0						
0A44	0C00	00C0	0070	00F0						
0A45	4809	EC5C	1000	00F0						
0A46	633C	0000	0000	00F0						


```

0A97 4829 0003 0C7C 00F0 03559600 0
0A98 4829 0003 0C26 00F0 03559600 0
0A99 4809 0C40 2000 00F0 03540C00 0
0A9A 4809 0003 9070 00F0 03541600 0
0A9B 480C 0073 0000 0000 03542C00 0
0A9C 4809 0156 0B00 00F0 03543000 0
0A9D 31C0 0070 0C70 006F 03544000 0
0A9E 2F30 0003 0C70 0060 03545000 0
0A9F 4809 0C40 9070 00F0 03546000 0
0AA0 9000 0000 0000 0C10 03547000 0
0AA1 4809 0000 0000 00F0 03548000 0
0AA2 5C09 0002 1070 00F0 03549000 0
0AA3 4809 0000 1070 00F0 03550000 0
0AA4 0000 0000 0000 0050 03551000 0
0AA5 4809 0C5C 0070 00F0 03552000 0
0AA6 4809 0000 0000 00F0 03553000 0
0AA7 4809 0C40 0000 00F0 03554000 0
0AA8 4809 0C41 0070 00F0 03555000 0
0AA9 0000 0000 0000 00F0 03556000 0
0ABA 2F50 0000 0B00 00F0 03557000 0
0AAC 0000 0000 0000 00F0 03558000 0
0AAB 2000 0000 0000 006C 03559000 0
0AAC 56EC 0000 0000 0060 03559600 0

OP113:
0A5E 5700 0000 0C7C 0060 03563000 0
0A5F 4809 0C41 0030 00F0 03564000 0
0A60 0C0C 0000 0C70 003C 03565000 0
0A61 4809 0003 9070 00F0 03566000 0
0A62 30FC 0000 0000 0080 03567000 0
0A63 4809 0156 0B30 00F0 03568000 0
0A64 51C0 0000 0C30 0060 03569000 0
0A65 2F30 0000 0C00 006C 03570000 0
0A66 4809 0C40 0030 00F0 03571000 0
0A67 4809 0C43 001C 00F0 03572000 0
0A68 51EC 0003 0C7C 0060 03573000 0
0A69 56EC 0000 0030 0040 03574000 0
0A69 56EC 0000 0030 0040 03575000 0
0A69 56EC 0000 0030 0040 03576000 0
0A69 56EC 0000 0030 0040 03577000 0
0A69 56EC 0000 0030 0040 03578000 0
0A69 56EC 0000 0030 0040 03579000 0
0A69 56EC 0000 0030 0040 03580000 0
0A69 56EC 0000 0030 0040 03581000 0
0A69 56EC 0000 0030 0040 03582000 0
0A69 56EC 0000 0030 0040 03583000 0
0A69 56EC 0000 0030 0040 03584000 0
0A69 56EC 0000 0030 0040 03585000 0
0A69 56EC 0000 0030 0040 03586000 0
0A69 56EC 0000 0030 0040 03587000 0
0A69 56EC 0000 0030 0040 03588000 0
0A69 56EC 0000 0030 0040 03589000 0
0A69 56EC 0000 0030 0040 03590000 0
0A69 56EC 0000 0030 0040 03591000 0
0A69 56EC 0000 0030 0040 03592000 0
0A69 56EC 0000 0030 0040 03593000 0
0A69 56EC 0000 0030 0040 03594000 0
0A69 56EC 0000 0030 0040 03595000 0
0A69 56EC 0000 0030 0040 03596000 0
0A69 56EC 0000 0030 0040 03597000 0

OP120:
0A6F 1400 0000 0000 0040 03598000 0
0A70 14C0 0000 0C70 0040 03599000 0
0A71 140C 0000 0C00 0040 03599600 0
0A72 14EC 00C3 003C 0040 03599600 0
0A73 4809 0C40 0B00 00F0 03599600 0
0A74 50F0 0000 0C7C 0080 03599600 0
0A75 4809 2C56 0B00 00F0 03599600 0

```



```

0449 9809 0C52 0000 00FF 03559C00 0
0AAA 9809 00C0 0030 00FF 0366CC00 0
0AAB 23AC 00C0 0C30 00FF 03661100 0
0AAC 9809 E000 A09C 00FF 03662400 0
0AAD 00C0 00C0 7000 00FF 03663C00 0
0AAE 9809 CC40 209C 00FF 03664400 0
0AAB 9809 E000 809C 00FF 03665400 0
0CAB 9000 0000 0070 00FF 03666400 0
0AB1 9809 2C5E 0030 00FF 03667400 0
0AB2 00C0 00C0 0030 00FF 03668000 0
0AB3 51CC 00C0 009C 00FF 03669000 0
0AB4 2F3C 00C0 007C 00FF 03670000 0
0AB5 9809 0F41 007C 00FF 03671000 0
0AB6 00C0 00C0 0C30 00FF 03672000 0
0AB7 9809 0C41 1070 00FF 03673000 0
0AB8 00C0 00C0 009C 00FF 03674000 0
0AB9 9809 2F5C 001C 00FF 03675000 0
0ABA 2F3C 00C0 007C 00FF 03676000 0
0ABB 9809 C000 009C 00FF 03677000 0
0ABC 9809 CC40 009C 00FF 03678000 0
0ABD 9809 C000 0030 00FF 03679000 0
0ABE 2000 00C0 8C30 00FF 03680000 0
0ABF 9809 C000 0030 00FF 03681000 0
0AC0 00C0 00C0 009C 00FF 03682000 0
0AC1 9809 00C0 009C 00FF 03683000 0
0AC2 9809 0F43 801C 00FF 03684000 0
0AC3 0000 00C0 0C00 00FF 03685000 0
0AC4 2F5C 00C0 009C 00FF 03686000 0
0AC5 9809 2F5C 001C 00FF 03687000 0
0AC6 001C 00C0 0C30 00FF 03688000 0
0AC7 9809 C001 209C 00FF 03689000 0
0AC8 9809 C0C0 8C30 00FF 03690000 0
0AC9 2F50 00C0 009C 00FF 03691000 0
0ACA 5650 00C0 009C 00FF 03692000 0
0ACB 570C 00C0 0C30 00FF 03693000 0
0ACC 9809 0C40 9030 00FF 03694000 0
0ACD 00C0 00C0 0070 00FF 03695000 0
0ACE 9809 00C0 0070 00FF 03696000 0
0ACF 9809 E15E 0030 00FF 03697000 0
0AD0 00C0 00C0 0C30 00FF 03698000 0
0AD1 51CC 00C0 009C 00FF 03699000 0
0AD2 2F30 00C0 0030 00FF 03700000 0
0AD3 9809 0C40 0030 00FF 03701000 0
0AD4 9809 0F40 107C 00FF 03702000 0
0AD5 9809 0C41 0010 00FF 03703000 0
0AD6 0000 00C0 0C30 00FF 03704000 0
0AD7 51CC 00C0 009C 00FF 03705000 0
0AD8 9809 2F5C 703C 00FF 03706000 0
0AD9 0010 00C0 009C 00FF 03707000 0
0ADA 9809 E0C0 001C 00FF 03708000 0
0ADB 2F30 00C0 009C 00FF 03709000 0
0ADC 9809 0C40 003C 00FF 03710000 0
0ADD 9809 C0C1 001C 00FF 03711000 0
0ADE 00C0 0C00 009C 00FF 03712000 0
0ADF 51CC 00C0 009C 00FF 03713000 0
0AE0 5650 00C0 009C 00FF 03714000 0
0AE1 570C 00C0 0C30 00FF 03715000 0
0AE2 9809 0C40 9030 00FF 03716000 0
0AE3 00C0 00C0 0070 00FF 03717000 0
0AE4 9809 00C0 0070 00FF 03718000 0
0AE5 9809 E15E 0030 00FF 03719000 0
0AE6 00C0 00C0 0C30 00FF 03720000 0
0AE7 51CC 00C0 009C 00FF 03721000 0
0AE8 2F30 00C0 0030 00FF 03722000 0
0AE9 9809 0C40 0030 00FF 03723000 0
0AEA 9809 0F40 107C 00FF 03724000 0
0AEB 9809 0C41 0010 00FF 03725000 0
0AEC 0000 00C0 0C30 00FF 03726000 0
0AED 51CC 00C0 009C 00FF 03727000 0
0AEE 9809 2F5C 703C 00FF 03728000 0
0AEO 0010 00C0 009C 00FF 03729000 0
0AE1 9809 E0C0 001C 00FF 03730000 0
0AE2 2F30 00C0 009C 00FF 03731000 0
0AE3 9809 0C40 003C 00FF 03732000 0
0AE4 9809 C0C1 001C 00FF 03733000 0
0AE5 00C0 0C00 009C 00FF 03734000 0
0AE6 51CC 00C0 009C 00FF 03735000 0
0AE7 5650 00C0 009C 00FF 03736000 0
0AE8 570C 00C0 0C30 00FF 03737000 0
0AE9 9809 0C40 9030 00FF 03738000 0
0AEA 00C0 00C0 0070 00FF 03739000 0
0AEB 9809 00C0 0070 00FF 03740000 0
0AEC 9809 E15E 0030 00FF 03741000 0
0AED 00C0 00C0 0C30 00FF 03742000 0
0AEE 51CC 00C0 009C 00FF 03743000 0
0AEO 2F30 00C0 0030 00FF 03744000 0
0AE1 9809 0C40 0030 00FF 03745000 0
0AE2 9809 0F40 107C 00FF 03746000 0
0AE3 9809 0C41 0010 00FF 03747000 0
0AE4 0000 00C0 0C30 00FF 03748000 0
0AE5 51CC 00C0 009C 00FF 03749000 0
0AE6 2F30 00C0 0030 00FF 03750000 0
0AE7 9809 0C40 0030 00FF 03751000 0
0AE8 9809 0F40 107C 00FF 03752000 0
0AE9 9809 0C41 0010 00FF 03753000 0
0AEO 0000 00C0 0C30 00FF 03754000 0

```

0P1231

```

04CB 570C 00C0 0C30 00FF 03755000 0
0ACC 9809 0C40 9030 00FF 03756000 0
0ACD 00C0 00C0 0070 00FF 03757000 0
0ACE 9809 00C0 0070 00FF 03758000 0
0ACF 9809 E15E 0030 00FF 03759000 0
0AD0 00C0 00C0 0C30 00FF 03760000 0
0AD1 51CC 00C0 009C 00FF 03761000 0
0AD2 2F30 00C0 0030 00FF 03762000 0
0AD3 9809 0C40 0030 00FF 03763000 0
0AD4 9809 0F40 107C 00FF 03764000 0
0AD5 9809 0C41 0010 00FF 03765000 0
0AD6 0000 00C0 0C30 00FF 03766000 0
0AD7 51CC 00C0 009C 00FF 03767000 0
0AD8 9809 2F5C 703C 00FF 03768000 0
0AD9 0010 00C0 009C 00FF 03769000 0
0ADA 9809 E0C0 001C 00FF 03770000 0
0ADB 2F30 00C0 009C 00FF 03771000 0
0ADC 9809 0C40 003C 00FF 03772000 0
0ADD 9809 C0C1 001C 00FF 03773000 0
0ADE 00C0 0C00 009C 00FF 03774000 0
0ADF 51CC 00C0 009C 00FF 03775000 0
0AE0 5650 00C0 009C 00FF 03776000 0
0AE1 570C 00C0 0C30 00FF 03777000 0
0AE2 9809 0C40 9030 00FF 03778000 0
0AE3 00C0 00C0 0070 00FF 03779000 0
0AE4 9809 00C0 0070 00FF 03780000 0
0AE5 9809 E15E 0030 00FF 03781000 0
0AE6 00C0 00C0 0C30 00FF 03782000 0
0AE7 51CC 00C0 009C 00FF 03783000 0
0AE8 2F30 00C0 0030 00FF 03784000 0
0AE9 9809 0C40 0030 00FF 03785000 0
0AEA 9809 0F40 107C 00FF 03786000 0
0AEB 9809 0C41 0010 00FF 03787000 0
0AEC 0000 00C0 0C30 00FF 03788000 0
0AED 51CC 00C0 009C 00FF 03789000 0
0AEO 2F30 00C0 0030 00FF 03790000 0
0AE1 9809 0C40 0030 00FF 03791000 0
0AE2 9809 0F40 107C 00FF 03792000 0
0AE3 9809 0C41 0010 00FF 03793000 0
0AEO 0000 00C0 0C30 00FF 03794000 0

```


OPCODE 13:

00AE1	5F90	0000	0000	C060
00AE2	4809	C640	0014	C0F0
00AE3	14FC	0C00	0000	C0C0
00AE4	4809	00C0	0090	00FC
00AE5	4824	0000	C030	00F0

```
XFCODE - 1 = CPR
A2 + AMPCR = AMPCR
OP13F - 1 = AMPCR
STEP
EXEC
```

OP13F1

0AE6	1500	0000	0000	0040
0AE7	3000	0000	0000	0040
0AE8	1510	0000	0030	0040
0AE9	1520	0000	0030	0040

OP130	-	1	=	MPCR
FAULT	-	1	=	MPCR
OP132	-	1	=	MPCR
OP133	-	1	=	MPCR

OP130:

AREA	4809	0C40	8B30	00F0
AREA	80FC	00C0	0C30	0080
AREA	4809	2C5E	8B0C	00F0
AREA	51C0	0C00	0C30	0060
AREA	2F30	0000	0C30	0060
AREA	4809	0C41	2030	00F0
AREA	0010	0003	0000	00A0
AREA	4809	0F41	0C20	00F0
AREA	0000	00C3	0000	0030
AREA	4809	2F5C	0C1C	00FC
AREA	2F30	0C00	0000	0060
AREA	4809	CC5C	2030	00F0

```

0 R = 0
LIT 4 = SAR1 15 = LIT
LIT AND 0 = 0
REGSTACK - 1 = CPCR
IF INPUT - 1 = CPCR
    R L = A2
    COMP 16 = SAR1 1 = L
    BMR L = BR1
    COMP 8 = SAR
    LIT OR BMR = MAR2
    IF INPUT - 1 = CPCR
        COMP 8 = A2
        OR R = A2

```

OP132:

0800 4809 2056 0830 00FN

LIT AND B = B

183

[illegible]

OP133:

CPB39	6330	0000	0000	0060
CPB38	3809	0C41	0C00	0C0C
CPB37	0000	0000	0C00	003C
CPB36	3609	E156	2000	00FC
CPB35	80FC	0000	0000	0080
CPB34	3409	E000	9C00	0C0C
CPB33	3309	E155	0B90	00F0
CPB32	4809	CC5E	0B3C	00F0
CPB31	4419	0C46	0600	00F0
CPB2	4809	2C80	0B00	00F0
CPB1	0110	0C0C	0000	00E0
CPB4	4809	0C5F	1D00	00F0
CPB5	000C	0C00	0070	0020


```

00A8 4899 00C3 00C0 00F0 03959000 0
00A9 5700 0000 0000 0060 03966000 0
00AA 4809 0041 0020 0060 03966100 0
00AB 4800 00C3 000C 00C0 03966200 0
00AC 4809 0000 0000 00F0 03966300 0
00AD 4809 0000 0000 0060 03966400 0
00AE 4809 00C3 000C 00C0 03966500 0
00AF 4809 00C3 000C 00C0 03966600 0
00B0 4809 00C3 000C 00C0 03966700 0
00B1 4809 00C3 000C 00C0 03966800 0
00B2 4809 00C3 000C 00C0 03966900 0
00B3 4809 00C3 000C 00C0 03967000 0
00B4 4809 00C3 000C 00C0 03967100 0
00B5 4809 00C3 000C 00C0 03967200 0
00B6 4809 00C3 000C 00C0 03967300 0
00B7 4809 00C3 000C 00C0 03967400 0
00B8 4809 00C3 000C 00C0 03967500 0
00B9 4809 00C3 000C 00C0 03967600 0
00BA 4809 00C3 000C 00C0 03967700 0
00BB 4809 00C3 000C 00C0 03967800 0
00BC 4809 00C3 000C 00C0 03967900 0
00BD 4809 00C3 000C 00C0 03968000 0
00BE 4809 00C3 000C 00C0 03968100 0
00BF 4809 00C3 000C 00C0 03968200 0
00C0 4809 00C3 000C 00C0 03968300 0
00C1 4809 00C3 000C 00C0 03968400 0
00C2 4809 00C3 000C 00C0 03968500 0
00C3 4809 00C3 000C 00C0 03968600 0
00C4 4809 00C3 000C 00C0 03968700 0
00C5 4809 00C3 000C 00C0 03968800 0
00C6 4809 00C3 000C 00C0 03968900 0
00C7 4809 00C3 000C 00C0 03969000 0
00C8 4809 00C3 000C 00C0 03969100 0
00C9 4809 00C3 000C 00C0 03969200 0
00CA 4809 00C3 000C 00C0 03969300 0
00CB 4809 00C3 000C 00C0 03969400 0
00CC 4809 00C3 000C 00C0 03969500 0
00CD 5700 0000 0000 0060 03969600 0
00CE 4809 0040 0030 0060 03969700 0
00CF 15A0 00C3 0000 00C0 03969800 0
00D0 4809 00C3 0000 00C0 03969900 0
00D1 4824 0000 0000 00C0 03970000 0
00D2 15B0 00C3 0000 0040 03970100 0
00D3 15C0 0000 0000 0040 03970200 0
00D4 15D0 0000 0000 0040 03970300 0
00D5 15E0 0000 0000 0040 03970400 0
00D6 2380 0000 0000 0060 03970500 0
00D7 4809 0040 0030 0060 03970600 0
00D8 4809 00C3 0000 0060 03970700 0
00D9 4809 00C3 0000 0060 03970800 0
00DA 4809 00C3 0000 0060 03970900 0
00DB 4809 00C3 0000 0060 03971000 0
00DC 4809 00C3 0000 0060 03971100 0
00DD 4809 00C3 0000 0060 03971200 0
00DE 4809 00C3 0000 0060 03971300 0
00DF 4809 00C3 0000 0060 03971400 0
00E0 4809 00C3 0000 0060 03971500 0
00E1 4809 00C3 0000 0060 03971600 0
00E2 4809 00C3 0000 0060 03971700 0
00E3 4809 00C3 0000 0060 03971800 0
00E4 4809 00C3 0000 0060 03971900 0
00E5 4809 00C3 0000 0060 03972000 0
00E6 4809 00C3 0000 0060 03972100 0
00E7 4809 00C3 0000 0060 03972200 0
00E8 4809 00C3 0000 0060 03972300 0
00E9 4809 00C3 0000 0060 03972400 0
00EA 4809 00C3 0000 0060 03972500 0
00EB 4809 00C3 0000 0060 03972600 0
00EC 4809 00C3 0000 0060 03972700 0
00ED 4809 00C3 0000 0060 03972800 0
00EE 4809 00C3 0000 0060 03972900 0
00EF 4809 00C3 0000 0060 03973000 0
00F0 4809 00C3 0000 0060 03973100 0
00F1 4809 00C3 0000 0060 03973200 0
00F2 4809 00C3 0000 0060 03973300 0
00F3 4809 00C3 0000 0060 03973400 0
00F4 4809 00C3 0000 0060 03973500 0
00F5 4809 00C3 0000 0060 03973600 0
00F6 4809 00C3 0000 0060 03973700 0
00F7 4809 00C3 0000 0060 03973800 0
00F8 4809 00C3 0000 0060 03973900 0
00F9 4809 00C3 0000 0060 03974000 0
00FA 4809 00C3 0000 0060 03974100 0
00FB 4809 00C3 0000 0060 03974200 0
00FC 4809 00C3 0000 0060 03974300 0
00FD 4809 00C3 0000 0060 03974400 0
00FE 4809 00C3 0000 0060 03974500 0
00FF 4809 00C3 0000 0060 03974600 0

```

```

SET LC2
RNFIELD - 1 = CPCR
D L = BR1
COMP 8 = SAR
A3 = B
CONTENTISRA - 1 = CPCR
LIT AND B = MIR
255 = LIT
CONTENTISRH - 1 = CPCR
D = MIR, BHI
D = A3
EOUTPUT - 1 = CPCR
B = A3
ASR
BHAR = DP2
EULIN - 1 = CPCR
IF LC2 THEN STEP (LSE SNIP IF LC2, PUT BYTE INTO LS BYTE
LS143 - 1 = CPCR
B L = B, CSAR
COMP 24 = SAR
B R = B
A3 L = A3
COMP 8 = SAR
A3 OR B = MIR
CHAR L = BR2
COMP 8 = SAR
EULOUT - 1 = CPCR
B R = B, CSAR
B L = B
A3 OR B = MIR
CHAR L = BR2
EULOUT - 1 = CPCR
OPCODE - 1 = MPCR
OP15C - 1 = MPCR
OP151 - 1 = MPCR
OP152 - 1 = MPCR
OP153 - 1 = MPCR
EXEC
OP15F:
OP15C - 1 = MPCR
OP151 - 1 = MPCR
OP152 - 1 = MPCR
OP153 - 1 = MPCR
EXEC
OP150:
CONTENTISRH - 1 = CPCR
E = MIR
A3 = B
CONTENTISRA - 1 = CPCR
E = A2
A2 L = A2
COMP 16 = SAR, A3 = LIT
A2 OR B = A2, BHI
A2 = (R(A3))/(R(A3)), (R(H)) INTO E

```

```

03959000 0
03966000 0
03966100 0
03966200 0
03966300 0
03966400 0
03966500 0
03966600 0
03966700 0
03966800 0
03966900 0
03967000 0
03967100 0
03967200 0
03967300 0
03967400 0
03967500 0
03967600 0
03967700 0
03967800 0
03967900 0
03968000 0
03968100 0
03968200 0
03968300 0
03968400 0
03968500 0
03968600 0
03968700 0
03968800 0
03968900 0
03969000 0
03969100 0
03969200 0
03969300 0
03969400 0
03969500 0
03969600 0
03969700 0
03969800 0
03969900 0
03970000 0
03970100 0
03970200 0
03970300 0
03970400 0
03970500 0
03970600 0
03970700 0
03970800 0
03970900 0
03971000 0
03971100 0
03971200 0
03971300 0
03971400 0
03971500 0
03971600 0
03971700 0
03971800 0
03971900 0
03972000 0
03972100 0
03972200 0
03972300 0
03972400 0
03972500 0
03972600 0
03972700 0
03972800 0
03972900 0
03973000 0
03973100 0
03973200 0
03973300 0
03973400 0
03973500 0
03973600 0
03973700 0
03973800 0
03973900 0
03974000 0
03974100 0
03974200 0
03974300 0
03974400 0
03974500 0
03974600 0
03974700 0
03974800 0
03974900 0
03975000 0
03975100 0
03975200 0
03975300 0
03975400 0
03975500 0
03975600 0
03975700 0
03975800 0
03975900 0
03976000 0
03976100 0
03976200 0
03976300 0
03976400 0
03976500 0
03976600 0
03976700 0
03976800 0
03976900 0
03977000 0
03977100 0
03977200 0
03977300 0
03977400 0
03977500 0
03977600 0
03977700 0
03977800 0
03977900 0
03978000 0
03978100 0
03978200 0
03978300 0
03978400 0
03978500 0
03978600 0
03978700 0
03978800 0
03978900 0
03979000 0
03979100 0
03979200 0
03979300 0
03979400 0
03979500 0
03979600 0
03979700 0
03979800 0
03979900 0
03980000 0
03980100 0
03980200 0
03980300 0
03980400 0
03980500 0
03980600 0
03980700 0
03980800 0
03980900 0
03981000 0
03981100 0
03981200 0
03981300 0
03981400 0
03981500 0
03981600 0
03981700 0
03981800 0
03981900 0
03982000 0
03982100 0
03982200 0
03982300 0
03982400 0
03982500 0
03982600 0
03982700 0
03982800 0
03982900 0
03983000 0
03983100 0
03983200 0
03983300 0
03983400 0
03983500 0
03983600 0
03983700 0
03983800 0
03983900 0
03984000 0
03984100 0
03984200 0
03984300 0
03984400 0
03984500 0
03984600 0
03984700 0
03984800 0
03984900 0
03985000 0
03985100 0
03985200 0
03985300 0
03985400 0
03985500 0
03985600 0
03985700 0
03985800 0
03985900 0
03986000 0
03986100 0
03986200 0
03986300 0
03986400 0
03986500 0
03986600 0
03986700 0
03986800 0
03986900 0
03987000 0
03987100 0
03987200 0
03987300 0
03987400 0
03987500 0
03987600 0
03987700 0
03987800 0
03987900 0
03988000 0
03988100 0
03988200 0
03988300 0
03988400 0
03988500 0
03988600 0
03988700 0
03988800 0
03988900 0
03989000 0
03989100 0
03989200 0
03989300 0
03989400 0
03989500 0
03989600 0
03989700 0
03989800 0
03989900 0
03990000 0
03990100 0
03990200 0
03990300 0
03990400 0
03990500 0
03990600 0
03990700 0
03990800 0
03990900 0
03991000 0
03991100 0
03991200 0
03991300 0
03991400 0
03991500 0
03991600 0
03991700 0
03991800 0
03991900 0
03992000 0
03992100 0
03992200 0
03992300 0
03992400 0
03992500 0
03992600 0
03992700 0
03992800 0
03992900 0
03993000 0
03993100 0
03993200 0
03993300 0
03993400 0
03993500 0
03993600 0
03993700 0
03993800 0
03993900 0
03994000 0
03994100 0
03994200 0
03994300 0
03994400 0
03994500 0
03994600 0
03994700 0
03994800 0
03994900 0
03995000 0
03995100 0
03995200 0
03995300 0
03995400 0
03995500 0
03995600 0
03995700 0
03995800 0
03995900 0
03996000 0
03996100 0
03996200 0
03996300 0
03996400 0
03996500 0
03996600 0
03996700 0
03996800 0
03996900 0
03997000 0
03997100 0
03997200 0
03997300 0
03997400 0
03997500 0
03997600 0
03997700 0
03997800 0
03997900 0
03998000 0
03998100 0
03998200 0
03998300 0
03998400 0
03998500 0
03998600 0
03998700 0
03998800 0
03998900 0
03999000 0
03999100 0
03999200 0
03999300 0
03999400 0
03999500 0
03999600 0
03999700 0
03999800 0
03999900 0
04000000 0
04000100 0
04000200 0
04000300 0
04000400 0
04000500 0
04000600 0
04000700 0
04000800 0
04000900 0
04001000 0
04001100 0
04001200 0
04001300 0
04001400 0
04001500 0
04001600 0
04001700 0
04001800 0
04001900 0
04002000 0
04002100 0
04002200 0
04002300 0
04002400 0
04002500 0
04002600 0
04002700 0
04002800 0
04002900 0
04003000 0
04003100 0
04003200 0
04003300 0
04003400 0
04003500 0
04003600 0
04003700 0
04003800 0
04003900 0
04004000 0
04004100 0
04004200 0
04004300 0
04004400 0
04004500 0
04004600 0
04004700 0
04004800 0
04004900 0
04005000 0
04005100 0
04005200 0
04005300 0
04005400 0
04005500 0
04005600 0
04005700 0
04005800 0
04005900 0
04006000 0
04006100 0
04006200 0
04006300 0
04006400 0
04006500 0
04006600 0
04006700 0
04006800 0
04006900 0
04007000 0
04007100 0
04007200 0
04007300 0
04007400 0
04007500 0
04007600 0
04007700 0
04007800 0
04007900 0
04008000 0
04008100 0
04008200 0
04008300 0
04008400 0
04008500 0
04008600 0
04008700 0
04008800 0
04008900 0
04009000 0
04009100 0
04009200 0
04009300 0
04009400 0
04009500 0
04009600 0
04009700 0
04009800 0
04009900 0
04010000 0
04010100 0
04010200 0
04010300 0
04010400 0
04010500 0
04010600 0
04010700 0
04010800 0
04010900 0
04011000 0
04011100 0
04011200 0
04011300 0
04011400 0
04011500 0
04011600 0
04011700 0
04011800 0
04011900 0
04012000 0
04012100 0
04012200 0
04012300 0
04012400 0
04012500 0
04012600 0
04012700 0
04012800 0
04012900 0
04013000 0
04013100 0
04013200 0
04013300 0
04013400 0
04013500 0
04013600 0
04013700 0
04013800 0
04013900 0
04014000 0
04014100 0
04014200 0
04014300 0
04014400 0
04014500 0
04014600 0
04014700 0
04014800 0
04014900 0
04015000 0
04015100 0
04015200 0
04015300 0
04015400 0
04015500 0
04015600 0
04015700 0
04015800 0
04015900 0
04016000 0
04016100 0
04016200 0
04016300 0
04016400 0
04016500 0
04016600 0
04016700 0
04016800 0
04016900 0
04017000 0
04017100 0
04017200 0
04017300 0
04017400 0
04017500 0
04017600 0
04017700 0
04017800 0
04017900 0
04018000 0
04018100 0
04018200 0
04018300 0
04018400 0
04018500 0
04018600 0
04018700 0
04018800 0
04018900 0
04019000 0
04019100 0
04019200 0
04019300 0
04019400 0
04019500 0
04019600 0
04019700 0
04019800 0
04019900 0
04020000 0

```


[illegible]

OP161:

```

233C 0000 C070 0060
4309 0541 002C 0058
002C CFC0 0000 0080
4809 2540 004C 00F0
4809 0F41 001C 00F0
51E0 00C5 0000 C06F
48C9 E800 8000 0000
80FC 0000 0000 0080
48C9 2555 0020 00F0
51C0 0000 0070 0060
4809 EFC0 E820 00C0
2280 0000 0000 C060
4809 0C40 009C 00F0
4809 00C0 0070 20F0
4809 0F43 0010 00F0
61E0 C0C0 0000 006F
4809 0F47 0020 00F0
00C0 00C0 0000 C03C
4809 EFC0 E81C 00F0
001C 0000 0000 C0AC
4809 2F5C 001C 00F0
2F3C C0C0 0020 C060
4809 0C40 009C 00F0
4809 0F40 0020 20F0
4809 0F43 0010 00F0
61E0 C0C0 0000 C060
56E0 0000 C020 0040

OP162:
0072 4809 2C55 007C 00F0
00FC 0000 0000 00FC
0071 56E0 0000 C020 0040

CONTENISHR - 1 = CPCR
B L = BR1
COMP 8 = SAR1 2 = LIT
LIT + 8 = MIR
BMR L = BR2
CMLUOUT - 1 = CPCR
A5 R = B
L5 R = B
LIT AND R = B
REGSTACK - 1 = CPCR
A5 = B
LIT AND R = B
CMLUOUT - 1 = CPCR
A5 R = B
L5 R = B
LIT AND R = B
REGSTACK - 1 = CPCR
A5 = B
LIT AND R = B
CMLUOUT - 1 = CPCR
BMR = BR2
BMR + 1 L = BR1
COMP 8 = SAR
A5 R = SAR1 1 = LIT
LIT OR BMR = MAR2
EMLUOUT - 1 = CPCR
B = MIR
ASR
BMR = BR2
CMLUOUT - 1 = CPCR
BMR + 1 L = BR1
COMP 8 = SAR
A5 R = SAR1 1 = LIT
LIT OR BMR = MAR2
EMLUOUT - 1 = CPCR
B = MIR
ASR
BMR = BR2
CMLUOUT - 1 = CPCR
OPCODE - 1 = MPCR

% (R(M)) + 2 INTO R(M)
% Y* INTO ER1
% (R(M)) + 2 INTO R(M)
% (R(M)) + 2 INTO R(M)
% ISOLATE *A*
% (R(A)) INTO E
% REFERENCE BR1
% (R(A)) INTO Y*
% ADDR OF Y* + 1
% ADDR OF R(A)
% (R(A+1)
% (R(A+1)) INTO B
% REFERENCE BR1
% (R(A+1)) INTO Y* + 1
%
%
% PK TYPE ALG LEFT DOUBLE SHIFT
% SHIFT (R(A),R(A+1) LEFT Y (0-5) PLACES
% ZERO FILL, SET CC AND SET OVERFLOW BIT
% ISOLATE *M*

```

08162:


```

CC74 4809 0052 0000 00F0
CC75 4819 0000 0000 00F0
CC76 4839 0000 0000 00F0
CC77 4839 0041 0000 00F0
CC78 0000 0000 0000 00F0
CC79 4809 EC5C 1C2F 00F0
CC7A 633C 0000 0000 00F0
CC7B 4809 E0C0 AC30 00F0
CC7C 0000 0000 0000 00F0
CC7D 4809 CC40 C030 00F0
CC7E 4809 E0C0 903C 00F0
CC7F 30F0 0000 0000 00F0
CC80 4809 E155 0070 00F0
CC81 51C0 0000 0000 00F0
CC82 4809 0F41 0000 00F0
CC83 0000 0000 0000 00F0
CC84 2F3C 0000 0000 00F0
CC85 4809 0C41 1000 00F0
CC86 0000 0000 0000 00F0
CC87 4809 2F5C 0000 00F0
CC88 2F3C 0000 0000 00F0
CC89 4809 EC5C 2000 00F0
CC8A 4809 005E 0000 00F0
CC8B 4809 0000 0000 00F0
CC8C 4809 0000 0000 00F0
CC8D 51C9 0000 0000 00F0
CC8E 5020 0000 0000 00F0
CC8F 4809 005E 0000 00F0
CC90 49C9 C001 107F 00F0
CC91 2000 0000 0000 00F0
CC92 4809 E0C1 2000 00F0
CC93 0000 0000 0000 00F0
CC94 4809 C0C0 8030 00F0
CC95 2F50 0000 0000 00F0
CC96 4809 E0C0 803C 00F0
CC97 4809 0000 0000 00F0
CC98 4809 CF43 801C 00F0
CC99 0000 0000 0000 00F0
CC9A 2F5C 0000 0000 00F0
CC9B 66E0 0000 0000 00F0

CC9C 570C 0000 0000 006C
CC9D 4809 0C41 0020 00F0
CC9E 2F3C 0000 0000 00F0
CC9F 2F3C 0000 0000 00F0
CC9G 4809 2F40 0000 00F0
CC9H 002C 0000 0000 00F0
CC9I 2F50 0000 0000 00F0
CC9J 4809 E0C3 3000 00F0
CC9K 80FC 0000 0000 00F0
CC9L 4809 E155 0000 00F0
CC9M 51C0 0000 0000 00F0
CC9N 2F30 0000 0000 00F0
CC9O 4809 0C40 0000 00F0
CC9P 4809 2F5C 1C00 00F0
CC9Q 001C 0000 0000 00F0

OP163:
0 EOL B
1 TRUE THEN SKIP
2 INTRUSION - 1 = CPCR
3 (R(H)) INTO B
4 L = SAR
5 A3 OR B = A3
6 IFETCH - 1 = CPCR
7 A3 R = A2
8 A2 + B = MIR
9 16 = SAR
10 A3 R = A3
11 q = SAR, 15 = LIT
12 A3 AND LIT = B
13 REGSTACK - 1 = CPCR
14 BHAR L = BRL
15 COMP 8 = SAR
16 INPUT - 1 = CPCR
17 B L = A3
18 COMP 16 = SAR, 1 = LIT
19 LIT OR BHAR = MAR2
20 INPUT - 1 = CPCR
21 A3 OR B = A2, BHI
22 -B = SAR
23 A2 R = A3
24 NOT A3
25 INPUT - 1 = CPCR
26 NOT A3
27 INPUT - 1 = CPCR
28 -B = SAR
29 A2 L = A3, B, SET LCI
30 A3 L = A2
31 SEICCA - 1 = CPCR
32 COMP 16 = SAR
33 A2 R = MIR
34 OUTPUT - 1 = CPCR
35 A3 R = MIR
36 ASR
37 BHAR R = MAR2
38 B = SAR
39 OUTPUT - 1 = CPCR
40 OPCODE - 1 = MPCR
41 RXHFIELD - 1 = CPCR
42 B L = BRL
43 COMP 8 = SAR
44 CONTENTSRM 1 = CPCR
45 LIT = B
46 LIT = MIR
47 OUTPUT - 1 = CPCR
48 A3 R = A3
49 q = SAR, 15 = LIT
50 A3 AND LIT = B
51 REGSTACK - 1 = CPCR
52 INPUT - 1 = CPCR
53 B = MIR
54 LIT OR BHAR = A3
55 1 = LIT
56 RX TYPE STORE AND INDEX Y 2
57 (R(A),R(A+1)) INTO Y, Y+1
58 (R(H)) + 2 INTO R(H)
59 Y INTO B
60 TEMP STORAGE OF Y INTO BFI
61 (R(H)) INTO B
62 TEMP STORAGE OF Y INTO R(H)
63 (R(H)) + 2 INTO R(H)
64 ISOLATE *A*
65 ADDR OF R(A) INTO MAR2
66 (R(A)) INTO E
67 ISOLATE *A*
68 R(A+1)
69 R(A+1)
70 R(A+1)
71 R(A+1)
72 R(A+1)
73 R(A+1)
74 R(A+1)
75 R(A+1)
76 R(A+1)
77 R(A+1)
78 R(A+1)
79 R(A+1)
80 R(A+1)
81 R(A+1)
82 R(A+1)
83 R(A+1)
84 R(A+1)
85 R(A+1)
86 R(A+1)
87 R(A+1)
88 R(A+1)
89 R(A+1)
90 R(A+1)
91 R(A+1)
92 R(A+1)
93 R(A+1)
94 R(A+1)
95 R(A+1)
96 R(A+1)
97 R(A+1)
98 R(A+1)
99 R(A+1)
100 R(A+1)

```



```

0P1711
PCDE 2360 00C3 00D0 0060
0CDF 4809 0C41 0010 00F0
0CE0 0000 0000 0000 0030
0CE1 4809 00C0 00D0 00F0
0CE2 61E0 00C0 00D0 0060
0CE3 66E0 0000 0000 0040

0P1712
PCDE 4809 2C56 00C0 00F0
0CE4 00C0 00C0 00D0 00E0
0CE5 4809 0C52 00D0 00F0
0CE6 6819 0000 0000 0060
0CE7 236C 0000 00D0 00F0
0CE8 4809 0C41 00D0 00F0
0CE9 4809 0C41 00D0 00F0
0CEA 0000 00C0 00D0 00F0
0CEB 4809 EC5C 1C30 00F0
0CEC 6330 00C0 0000 0060
0CED 4809 EC00 A000 00F0
0CEE 0000 00C0 00D0 00F0
0CEF 4809 CC40 27D0 00F0
0CF0 4809 EC00 9000 00F0
0CF1 50F0 00C0 00D0 00F0
0CF2 4809 E155 00D0 00F0
0CF3 51C0 00C0 00D0 0060
0CF4 4809 0F41 0C20 00F0
0CF5 001C 0000 00C0 0060
0CF6 2F3C 0000 00C0 0060
0CF7 4809 2F5C 001C 00F0
0CF8 4809 0C41 1C00 00F0
0CF9 0000 00C0 00D0 00F0
0CEA 2F50 00C0 00D0 0060
0CEB 4809 EC5C 30D0 00F0
0CEC 4809 C0C3 000C 00F0
0CEE 49C9 E0F1 9B20 00F0
0CFE 2000 00C0 00D0 0060
0CF0 4809 E0C1 2C30 00F0
0CF1 0000 00C0 00D0 00F0
0CF2 4809 C0C0 80D0 00F0
0CF3 2F5C 00C0 00D0 0060
0CF4 4809 0000 00D0 00F0
0CF5 4809 0F43 801C 00F0
0CF6 0000 00C0 00D0 00F0
0CF7 4809 E000 80D0 00F0
0CF8 0000 00C0 00D0 00F0
0CF9 2F50 00C0 00D0 0060
0CEA 66E0 00C0 00D0 0040

0P173:
0008 57C0 0000 00D0 0060
000C 4809 0C41 0010 00F0
0000 0000 00C0 00D0 0030
000E 4809 0C00 00C0 00F0

```

```

% RI TYPE 1 STORE ZERO, 0 INTO Y+
% (R(H)) INTO B
% A3 = (R(H))/INSTRUCTION
% Y+ INTO B
% (R(H)) INTO A2
% Y INTO A2
% ISOLATE "A"
% TEMP STORAGE OF R(A)
% (R(A)) INTO E
% (R(A)) INTO F5 WORD OF A3
% (R(A+1)) INTO B
% A3 = (R(A))/(R(A+1))
% COMP OF Y (0-5)
% PERFORM SHIFT
% SET THE COMBINATION BITS
% SHIFTED (R(A+1)) INTO HIF
% REFERENCE BR1
% RXHFIELD - 1 = CPCR
% B L = BR2
% COMP B = SAR
% C = MIR
% RXHFIELD - 1 = CPCR
% B L = BR2
% COMP B = SAR
% C = MIR

```


[illegible]


```

CE0E 4807 0000 0000 0000
CE0F 4809 0C40 8B30 C0FC
CE10 2F50 0000 0000 0060
CE11 2C00 0000 0000 0060
CE12 56E0 0000 0000 0060

CE13 22BC 0000 0000 0060
CE14 4809 0C41 2C30 00FC
CE15 00FC 0000 0000 0060
CE16 4809 0C40 8B30 C0FC
CE17 51CC 0000 0000 0060
CE18 1E7C 0000 0000 0060
CE19 0C40 0000 0000 0060
CE1A 0000 0000 0000 0060
CE1B 4809 0C40 8B30 C0FC
CE1C 4809 0C41 0B0C 00FC
CE1D 0000 0000 0000 0060
CE1E 4809 0C40 8B30 C0FC
CE1F 78C9 0000 0000 0060
CE20 1E7C 0000 0000 0060
CE21 4F1C 0000 0000 0060
CE22 4809 0C41 801C 00FC
CE23 0000 0000 0000 0060
CE24 48C9 0C40 8B30 C0FC
CE25 2F50 0000 0000 0060
CE26 200C 0000 0000 0060
CE27 56E0 0000 0000 0060

CE28 4809 2C55 0B3C 00FC
CE29 00FC 0000 0000 0060
CE2A 4809 0C52 0A30 00FC
CE2B 5819 0C00 0000 00FC
CE2C 238C 0000 0000 0060
CE2D 4809 0C41 0B0C 00FC
CE2E 4809 0C50 0000 0060
CE2F 4809 0C50 0000 0060
CE30 4809 0C40 8B30 C0FC
CE31 4809 0C40 8B30 C0FC
CE32 0000 0000 0000 0060
CE33 4809 0C41 0B0C 00FC
CE34 4809 0C03 0B3C 00FC
CE35 22BC 0000 0000 0060
CE36 4809 0C41 2C30 00FC
CE37 0000 0000 0000 0060
CE38 4809 0C40 8B30 C0FC
CE39 78C9 0000 0000 0060
CE3A 1E7C 0000 0000 0060
CE3B 4F10 0000 0000 0060
CE3C 4809 0000 0000 0060
CE3D 4809 0C40 8B30 C0FC
CE3E 0000 0000 0000 0060
CE3F 2F50 0000 0000 0060
CE40 2000 0000 0000 0060

CE41 4809 0C41 2C30 00FC
CE42 0000 0000 0000 0060
CE43 4809 0C40 8B30 C0FC
CE44 0000 0000 0000 0060
CE45 4809 0C41 0B0C 00FC
CE46 0000 0000 0000 0060
CE47 4809 0C40 8B30 C0FC
CE48 0000 0000 0000 0060
CE49 4809 0C41 2C30 00FC
CE4A 0000 0000 0000 0060
CE4B 4809 0C40 8B30 C0FC
CE4C 0000 0000 0000 0060
CE4D 4809 0C41 0B0C 00FC
CE4E 0000 0000 0000 0060
CE4F 4809 0C40 8B30 C0FC
CE50 0000 0000 0000 0060
CE51 4809 0C41 2C30 00FC
CE52 0000 0000 0000 0060
CE53 4809 0C40 8B30 C0FC
CE54 0000 0000 0000 0060
CE55 4809 0C41 0B0C 00FC
CE56 0000 0000 0000 0060
CE57 4809 0C40 8B30 C0FC
CE58 0000 0000 0000 0060
CE59 4809 0C41 2C30 00FC
CE60 0000 0000 0000 0060
CE61 4809 0C40 8B30 C0FC
CE62 0000 0000 0000 0060
CE63 4809 0C41 0B0C 00FC
CE64 0000 0000 0000 0060
CE65 4809 0C40 8B30 C0FC
CE66 0000 0000 0000 0060
CE67 4809 0C41 2C30 00FC
CE68 0000 0000 0000 0060
CE69 4809 0C40 8B30 C0FC
CE70 0000 0000 0000 0060
CE71 4809 0C41 0B0C 00FC
CE72 0000 0000 0000 0060
CE73 4809 0C40 8B30 C0FC
CE74 0000 0000 0000 0060
CE75 4809 0C41 2C30 00FC
CE76 0000 0000 0000 0060
CE77 4809 0C40 8B30 C0FC
CE78 0000 0000 0000 0060
CE79 4809 0C41 0B0C 00FC
CE80 0000 0000 0000 0060
CE81 4809 0C40 8B30 C0FC
CE82 0000 0000 0000 0060
CE83 4809 0C41 2C30 00FC
CE84 0000 0000 0000 0060
CE85 4809 0C40 8B30 C0FC
CE86 0000 0000 0000 0060
CE87 4809 0C41 0B0C 00FC
CE88 0000 0000 0000 0060
CE89 4809 0C40 8B30 C0FC
CE90 0000 0000 0000 0060
CE91 4809 0C41 2C30 00FC
CE92 0000 0000 0000 0060
CE93 4809 0C40 8B30 C0FC
CE94 0000 0000 0000 0060
CE95 4809 0C41 0B0C 00FC
CE96 0000 0000 0000 0060
CE97 4809 0C40 8B30 C0FC
CE98 0000 0000 0000 0060
CE99 4809 0C41 2C30 00FC
CEA0 0000 0000 0000 0060
CEA1 4809 0C40 8B30 C0FC
CEA2 0000 0000 0000 0060
CEA3 4809 0C41 0B0C 00FC
CEA4 0000 0000 0000 0060
CEA5 4809 0C40 8B30 C0FC
CEA6 0000 0000 0000 0060
CEA7 4809 0C41 2C30 00FC
CEA8 0000 0000 0000 0060
CEA9 4809 0C40 8B30 C0FC
CEAA 0000 0000 0000 0060
CEAB 4809 0C41 0B0C 00FC
CEAC 0000 0000 0000 0060
CEAD 4809 0C40 8B30 C0FC
CEAE 0000 0000 0000 0060
CEAF 4809 0C41 2C30 00FC
CEB0 0000 0000 0000 0060
CEB1 4809 0C40 8B30 C0FC
CEB2 0000 0000 0000 0060
CEB3 4809 0C41 0B0C 00FC
CEB4 0000 0000 0000 0060
CEB5 4809 0C40 8B30 C0FC
CEB6 0000 0000 0000 0060
CEB7 4809 0C41 2C30 00FC
CEB8 0000 0000 0000 0060
CEB9 4809 0C40 8B30 C0FC
CEBA 0000 0000 0000 0060
CEBB 4809 0C41 0B0C 00FC
CEBC 0000 0000 0000 0060
CEBD 4809 0C40 8B30 C0FC
CEBE 0000 0000 0000 0060
CEBF 4809 0C41 2C30 00FC
CEC0 0000 0000 0000 0060
CEC1 4809 0C40 8B30 C0FC
CEC2 0000 0000 0000 0060
CEC3 4809 0C41 0B0C 00FC
CEC4 0000 0000 0000 0060
CEC5 4809 0C40 8B30 C0FC
CEC6 0000 0000 0000 0060
CEC7 4809 0C41 2C30 00FC
CEC8 0000 0000 0000 0060
CEC9 4809 0C40 8B30 C0FC
CECA 0000 0000 0000 0060
CECB 4809 0C41 0B0C 00FC
CECC 0000 0000 0000 0060
CECD 4809 0C40 8B30 C0FC
CECE 0000 0000 0000 0060
CECF 4809 0C41 2C30 00FC
CED0 0000 0000 0000 0060
CED1 4809 0C40 8B30 C0FC
CED2 0000 0000 0000 0060
CED3 4809 0C41 0B0C 00FC
CED4 0000 0000 0000 0060
CED5 4809 0C40 8B30 C0FC
CED6 0000 0000 0000 0060
CED7 4809 0C41 2C30 00FC
CED8 0000 0000 0000 0060
CED9 4809 0C40 8B30 C0FC
CEDA 0000 0000 0000 0060
CEDB 4809 0C41 0B0C 00FC
CEDC 0000 0000 0000 0060
CEDD 4809 0C40 8B30 C0FC
CEDE 0000 0000 0000 0060
CEDF 4809 0C41 2C30 00FC
CEE0 0000 0000 0000 0060
CEE1 4809 0C40 8B30 C0FC
CEE2 0000 0000 0000 0060
CEE3 4809 0C41 0B0C 00FC
CEE4 0000 0000 0000 0060
CEE5 4809 0C40 8B30 C0FC
CEE6 0000 0000 0000 0060
CEE7 4809 0C41 2C30 00FC
CEE8 0000 0000 0000 0060
CEE9 4809 0C40 8B30 C0FC
CEEA 0000 0000 0000 0060
CEEB 4809 0C41 0B0C 00FC
CEEC 0000 0000 0000 0060
CEED 4809 0C40 8B30 C0FC
CEEF 4809 0C41 2C30 00FC
CEFA 0000 0000 0000 0060
CEFB 4809 0C40 8B30 C0FC
CEFC 0000 0000 0000 0060
CEFD 4809 0C41 0B0C 00FC
CEFE 0000 0000 0000 0060
CEFF 4809 0C40 8B30 C0FC

```



```

0E41 58EC 0C73 00C0 C040
OPCODE - 1 = MPCR

0P223:
0F42 570C 0000 00C0 C06C
CF43 48F9 C041 001C 00F0
0F44 000C 00C0 00C0 C03C
0E45 50E0 00C0 00C0 C060
0E46 4809 0C41 00C0 C0F0
0F47 0000 00C0 00C0 C02C
0E48 48F9 E000 00C0 C060
0F49 228F 00C0 00C0 C060
0F4A 48C9 0C41 2000 C0F0
0F4B 000C 00C0 00C0 C02C
0E4C 4809 C040 003C C0F0
0F4D 78C9 00C0 00C0 C060
0E4E 1E70 0000 0000 C060
0E4F 4F1C 00C0 00C0 C06C
0E50 4809 C003 0000 C060
0F51 4809 0C40 003C C0F0
0E52 2000 00C0 00C0 C020
0F53 2000 00C0 00C0 C020
0E54 2E5C 0000 00C0 C060
0F55 56E0 0003 0000 C040

OPCODE23: XFC00E - 1 = CPCR
A2 + AMPCR = AMPCR
0P23F - 1 = AMPCR
STEP
EXEC
0P23C - 1 = MPCR
0P231 - 1 = MPCR
FAULT - 1 = MPCR
0P233 - 1 = MPCR

0P230:
LIT AND B = R
15 = LIT
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
R L = A2
COMP 16 = SAR J 1 = LIT
LIT OR BVAR = MAR2
EINPUT - 1 = CPCR
A2 OR B = MIR
A3 R = B
4 = SAR J 15 = LIT
LIT AND R = 6
REGSTACK - 1 = CPCR
BHAR L = BR1
COMP 8 = SAR
EINPUT - 1 = CPCR
B L = A2
COMP 16 = SAR J 1 = LIT

0F56 5E90 00C0 00C0 C060
0E57 4809 C640 00C0 C0F0
0E58 1770 00C0 00C0 C0C0
0E59 4809 00C0 00C0 C0F0
0E5A 4824 00C0 00C0 C0F0
0E5B 1780 0000 00C0 C040
0F5C 179C 00C0 00C0 C040
0E5D 300C 00C0 00C0 C040
0E5E 17A0 0003 00C0 C040

0F5F 4809 2C55 00C0 C0FC
0E60 00F0 00C0 00C0 C0FC
0E61 51C0 00C0 00C0 C060
0E62 2F3C 00C0 00C0 C060
0E63 4809 0C41 20C0 C0FF
0E64 001C 0000 00C0 C040
0E65 48C9 2F5C 001C C0F0
0E66 2F3C 00C0 00C0 C06C
0E67 4809 CC5C 00C0 C0F0
0E68 4809 E0C0 00C0 C0FC
0E69 30FC 00C0 00C0 C08C
0E6A 4809 2C56 00C0 C0FC
0E6B 51C0 C000 00C0 C060
0E6C 48C9 C041 00C0 C0FC
0E6D 000C 00C0 00C0 C03C
0E6E 2F3C 00C0 00C0 C060
0E6F 4809 C041 20C0 C0FF
0F70 0010 00C0 00C0 C040

```



```

06DC 1700 0009 0000 0040
06DD 17EC 0003 0000 0040
06DE 17FC 0000 0000 0040

062401
06DE 2300 00C3 00C0 0060
06DF 4809 0C40 0030 00F0
06E0 4809 0C40 0030 00F0
06E1 4809 0C40 0030 00F0
06E2 220C 00C3 00C0 0060
06E3 4809 0C41 2030 00F0
06E4 00C0 00C3 0030 0020
06E5 4809 0C41 0030 00F0
06E6 2000 00C0 00C0 0060
06E7 4809 0C40 00C0 00F0
06E8 4809 0C5E 0030 00F0
06E9 7809 0C40 00C0 00F0
06EA 1E7C 00C3 0030 0060
06EB 4F10 00C0 0030 0060
06EC 56EC 00C0 0030 0060

062402
06ED 2300 00C3 0000 0060
06EE 4809 0C40 0030 00F0
06EF 2300 00C3 0000 0060
06F0 4809 0C41 0030 00F0
06F1 00C0 00C0 00C0 0030
06F2 4809 0C41 2030 00F0
06F3 00C0 00C3 0000 0020
06F4 6E0C 00C0 00C0 0060
06F5 4809 0C41 0030 00F0
06F6 00C0 00C0 00C0 0020
06F7 2C00 00C0 00C0 0060
06F8 4809 0C40 00C0 00C0
06F9 4849 0C5E 0030 00F0
06FA 7809 00C0 00C0 00F0
06FB 1E7C 00C0 00C0 0060
06FC 4F10 00C0 00C0 0060
06FD 56E0 00C0 00C0 0040

062403
06FE 4809 2C55 00C0 00F0
06FF 00C0 00C0 00C0 0060
0700 4809 0C52 00C0 00F0
0701 5819 00C3 00C0 00F0
0702 538C 00C3 0000 0060
0703 00C0 00C1 00C0 00F0
0704 00C0 00C1 00C0 0020
0705 4809 0C55 10C0 00F0
0706 5330 00C3 00C0 0060
0707 4809 0C40 00C0 00F0
0708 00C0 00C0 00C0 0020
0709 4809 0C41 00C0 00F0
070A 4809 0C40 00C0 00F0
070B 220C 00C0 00C0 0060
070C 4809 0C41 2030 00F0

062404
062401 - 1 = MPCR
062402 - 1 = MPCR
062403 - 1 = MPCR

CONTENISM - 1 = CPCR
B = MIR
A3 = R
CONTENISRA - 1 = CPCR
B L = A2, BHI
COMP 16 = SAR
B L = B, MIR
SETCC - 1 = CPCR
BHI
A2 - B = MIR, SET LC2
IF ADV THEN SET LC1
CARRY - 1 = CPCR
CHECKOV - 1 = CPCR
OPCODE - 1 = MPCR

CONTENISRA - 1 = CPCR
B = MIR
CONTENISM - 1 = CPCR
B L = BHI, BHI
COMP 16 = SAR
COMP 16 = SAR
FVULIN - 1 = CPCR
P L = B, MIR
COMP 16 = SAR
SETCC - 1 = CPCR
BHI
A2 - B = MIR, SET LC2
IF ADV THEN SET LC1
CARRY - 1 = CPCR
CHECKOV - 1 = CPCR
OPCODE - 1 = MPCR

IIT AND B = B
15 = LIT
OQL B
IF TRUE THEN SKIP
CONTENISM - 1 = CPCR
B L = R
COMP 16 = SAR
B L = B, MIR
FVULIN - 1 = CPCR
SETCC - 1 = CPCR
A3 = R = A2
16 = SAR
A2 + B = MIR
A3 = B
CONTENISRA - 1 = CPCR
B L = A2, BHI

TYPE RR COMPARE, (R(A));(R(H))
SET CC, CARRY AND OV BITS
X (R(H)) INTO B
X TEMP STORAGE OF (R(H))
X (R(A)) INTO B
X (R(H)) INTO MS WORD OF B, MIR
X SET THE CONDITION BITS
X (R(A)) INTO A2
X (Y*) INTO B
X (Y*) INTO MS WORD OF B, MIR
X SET THE CONDITION BITS
X (Y*) BACK INTO D
X (R(A)) INTO A2
X (Y*) INTO B
X (Y*) INTO MS WORD OF B, MIR
X SET CC, CARRY AND OV BITS
X TYPE RK COMPARE, (R(A));(Y)
X SET CC, CARRY AND OV BITS
X (R(H)) INTO B
X A3 = (R(Y))/INSTRUCTION
X Y* INTO B
X (R(H)) INTO A2
X Y INTO MS WORD OF MIR
X (R(A)) INTO B
X (R(A)) INTO MS WORD OF A2

```


[illegible]


```

0FA5 51CC 0000 0000 0000
0FA6 4809 EC01 403C 00F0
0FA7 0000 0000 0000 0000
0FA8 0000 0000 0000 0000
0FA9 2F50 0000 0000 0000
0FAA 56E0 0000 0000 0000

0P2611
CFAC 4809 CC40 880C 00F0
CFAC 3CFC 0000 0000 0000
CFAD 4809 2C55 680C 00F0
0FAE 51CC 0000 0000 0000
0FAF 4809 0F41 007C 00F0
0FB0 0000 0000 0000 0000
0FB1 4809 0F46 003C 00F0
0FB2 51CC 0000 0000 0000
0FB3 2F30 0000 0000 0000
0FB4 4809 0040 203C 00F0
0FB5 4809 1156 0000 00F0
0FB6 0000 0000 0000 0000
0FB7 51CC 0000 0000 0000
0FB8 2F30 0000 0000 0000
0FB9 4809 0C41 0040 00F0
0FBA 0000 0000 0000 0000
0FBC 51CC 0000 0000 0000
0FBD 4809 0000 0000 0000
0FBE 2000 0000 0000 0000
0FBF 4809 0000 0000 0000
0FC0 0000 0000 0000 0000
0FC1 4809 0000 0000 0000
0FC2 4809 0F47 801C 00F0
0FC3 0000 0000 0000 0000
0FC4 2F50 0000 0000 0000
0FC5 4809 0F45 0000 0000
0FC6 51CC 0000 0000 0000
0FC7 4809 E001 1000 00F0
0FC8 0000 0000 0000 0000
0FC9 4809 E0C0 803C 00F0
0FCA 2F50 0000 0000 0000
0FCB 66E0 0000 0000 0000

0P2621
CFCC 4809 2C55 680C 00F0
CFCD 4809 0000 0000 0000
CFCE 4809 0C52 0000 0000
CFCF 5810 0000 0000 0000
0F00 238C 0000 0000 0000
CF01 4809 0C41 0030 00F0
CF02 0000 0000 0000 0000
CF03 4809 EC5E 107C 00F0
CF04 4330 0000 0000 0000
0F05 4809 E000 A000 00F0
0F06 0000 0000 0000 0000
CF07 4809 CC40 2030 00F0
CF08 4809 E0C0 8800 00F0

0P2631
REGSTACK - 1 = CPCR
A3 L = A3
COMP 16 = SAR
A3 OUTPUT - 1 = CPCR
OPCODE - 1 = MPCR

R R = P
4 = SAR, 15 = LIT
LIT AND B = B
REGSTACK - 1 = CPCR
B MAR L = B#1
COMP 8 = SAR
B MAR + 1 = B
REGSTACK - 1 = CPCR
INPUT - 1 = CPCR
B = A2
A3 AND LIT = B
15 = LIT
REGSTACK - 1 = CPCR
INPUT - 1 = CPCR
F L = B#2
COMP 8 = SAR
F MULIN - 1 = CPCR
MULT - 1 = CPCR
A3 ECCA - 1 = CPCR
A3 R = MIR
16 = SAR
ASR
B MAR R = MAR2
B = SAR
EUIPUT - 1 = CPCR
B MAR + 1 = B
REGSTACK - 1 = CPCR
A3 L = A3
COMP 16 = SAR
A3 R = MIR
EUIPUT - 1 = CPCR
OPCODE - 1 = MPCR

% DI TYPE 2 MULTIPLY (INDIRECT), SET CC
% (RCA+1) X (Y*) = R(A),R(A+1), SET CC
% ISOLATE *A*
% R(A)
% R(A+1)
% R(A+1) INTO MAR2
% (RCA+1) INTO B
% (RCA+1) INTO A2
% ISOLATE *M*
% R(M) INTO MAR2
% Y* INTO E
% (Y*) INTO B
% MULTI (RCA+1) X (Y*)
% PRODUCT IN A3
% SET THE CONDITION BITS
% (RCA)
% DEF ERI
% R(A)
% R(A+1)
% R(A+1) INTO PAR2
% TYPE RK MULTIPLY (CONSTANT)
% (RCA+1) X Y = R(A),R(A+1), SET CC
% ISOLATE *M*
% CHECK FOR H = 0
% (R(C)) INTO B
% A3 = (R(C))/INSTRUCTION
% Y* INTO B
% (R(C)) INTO A2
% Y INTO B

```


[illegible]


```

1C77 4809 0C31 0030 00F0
1C78 0000 0000 0000 0000
1C79 2F30 0000 0000 0000
1C7A 4809 0C31 2000 00F0
1C7B 0000 0000 0000 0000
1C7C 4809 0C31 0000 00F0
1C7D 3100 0000 0000 0000
1C7E 2F30 0000 0000 0000
1C7F 4809 0C31 2000 00F0
1C80 49C9 0000 0000 0000
1C81 0000 0000 0000 0000
1C82 4809 0C31 0000 00F0
1C83 2800 0000 0000 0000
1C84 2800 0000 0000 0000
1C85 50AC 0000 0000 0000
1C86 2819 0000 0000 0000
1C87 5050 0000 0000 0000
1C88 4809 0C31 0000 00F0
1C89 2C00 0000 0000 0000
1C8A 4809 0C31 0000 00F0
1C8B 48C9 0000 0000 0000
1C8C 3000 0000 0000 0000
1C8D 2F30 0000 0000 0000
1C8E 4809 0C31 0000 00F0
1C8F 5050 0000 0000 0000
1C90 4809 0C31 0000 00F0
1C91 2F30 0000 0000 0000
1C92 56E9 0000 0000 0000

```

0P2731:

```

1C93 570C 00C3 0000 0000
1C94 4809 0C31 0010 00F0
1C95 0000 00C3 0000 0000
1C96 60E9 0000 0000 0000
1C97 4809 0C40 0030 00F0
1C98 4809 0C00 0000 0000
1C99 30C0 0000 0000 0000
1C9A 4809 2C50 0000 0000
1C9B 51CC 0000 0000 0000
1C9C 4809 0C31 0000 00F0
1C9D 0000 0000 0000 0000
1C9E 2F30 0000 0000 0000
1C9F 4809 0C31 2000 00F0
1CA0 0000 0000 0000 0000
1CA1 5050 0000 0000 0000
1CA2 2F30 0000 0000 0000
1CA3 4809 0C50 2000 00F0
1CA4 49C9 0000 0000 0000
1CA5 4809 0C31 0000 00F0
1CA6 4809 0C31 0000 00F0
1CA7 20C0 0000 0000 0000
1CA8 28C0 0000 0000 0000
1CA9 50AC 0000 0000 0000
1CAA 30AC 0000 0000 0000
1CAB 2819 0000 0000 0000
1CAC 5050 0000 0000 0000
1CAD 4809 0C31 0000 00F0
1CAE 2000 0000 0000 0000

```

```

BMAR L = BR1
COMP 0 = SAR
EINPUT - 1 = CPCR
B L = A2
COMP 16 = SAR
BMAR + 1 = B
REGSTACK - 1 = CPCR
EINPUT - 1 = CPCR
A2 OR B = A2
A3 R = B SET LC1
16 = SAR
B L = B
DIV - 1 = CPCR
IF LC1 THEN SET LC1
SETOVBIT - 1 = CPCR
IF LC1 THEN SKIP
CLEAROV - 1 = CPCR
A3 = B
SEITCCA - 1 = CPCR
A2 = MIR, ASR
BMAR R = MAR2
6 = SAR
EINPUT - 1 = CPCR
BMAR + 1 = B
REGSTACK - 1 = CPCR
A3 = MIR
EINPUT - 1 = CPCR
OPCODE - 1 = MPCR

RXHFIELD - 1 = CPCR
B L = BR2
COMP 0 = SAR
EMULIN - 1 = CPCR
B = MIR
A3 R = B
4 = SAR, 15 = LIT
LIT AND B = B
REGSTACK - 1 = CPCR
BMAR L = BR1
COMP 0 = SAR
EINPUT - 1 = CPCR
B L = A2
COMP 16 = SAR
BMAR + 1 = B
REGSTACK - 1 = CPCR
A2 OR B = A2
EMULI SET LC1
B L = B
COMP 16 = SAR
DIV - 1 = CPCR
IF LC1 THEN SET LC1
SETOVBIT - 1 = CPCR
IF LC1 THEN SKIP
CLEAROV - 1 = CPCR
A3 = B
SEITCCA - 1 = CPCR

```

```

05399C00 0
05400000 0
05400C00 0
05401000 0
05401400 0
05401800 0
05401C00 0
05402000 0
05402400 0
05402800 0
05402C00 0
05403000 0
05403400 0
05403800 0
05403C00 0
05404000 0
05404400 0
05404800 0
05404C00 0
05405000 0
05405400 0
05405800 0
05405C00 0
05406000 0
05406400 0
05406800 0
05406C00 0
05407000 0
05407400 0
05407800 0
05407C00 0
05408000 0
05408400 0
05408800 0
05408C00 0
05409000 0
05409400 0
05409800 0
05409C00 0
05410000 0
05410400 0
05410800 0
05411200 0
05411600 0
05412000 0
05412400 0
05412800 0
05413200 0
05413600 0
05414000 0
05414400 0
05414800 0
05415200 0
05415600 0
05416000 0
05416400 0
05416800 0
05417200 0
05417600 0
05418000 0
05418400 0
05418800 0
05419200 0
05419600 0
05420000 0
05420400 0
05420800 0
05421200 0
05421600 0
05422000 0
05422400 0
05422800 0
05423200 0
05423600 0
05424000 0
05424400 0
05424800 0
05425200 0
05425600 0
05426000 0
05426400 0
05426800 0
05427200 0
05427600 0
05428000 0
05428400 0
05428800 0
05429200 0
05429600 0
05430000 0
05430400 0
05430800 0
05431200 0
05431600 0
05432000 0
05432400 0
05432800 0
05433200 0
05433600 0
05434000 0
05434400 0
05434800 0
05435200 0
05435600 0
05436000 0
05436400 0
05436800 0
05437200 0
05437600 0
05438000 0
05438400 0
05438800 0
05439200 0
05439600 0
05440000 0
05440400 0
05440800 0
05441200 0
05441600 0
05442000 0
05442400 0
05442800 0
05443200 0
05443600 0
05444000 0
05444400 0
05444800 0
05445200 0
05445600 0
05446000 0
05446400 0
05446800 0
05447200 0
05447600 0
05448000 0
05448400 0
05448800 0
05449200 0
05449600 0
05450000 0
05450400 0
05450800 0
05451200 0
05451600 0
05452000 0
05452400 0
05452800 0
05453200 0
05453600 0
05454000 0
05454400 0
05454800 0
05455200 0
05455600 0
05456000 0
05456400 0
05456800 0
05457200 0

```



```

10C0 4809 0C41 0000 00F0 00518000 0
10C1 0000 0000 0000 0000 00519000 0
10C2 4809 EC5C 1000 00F0 00520000 0
10C3 633C 0000 0000 0000 00521000 0
10C4 4809 E000 AC00 0000 00522000 0
10C5 4809 E000 AC00 0000 00523000 0
10C6 4809 C640 0000 0000 00524000 0
10C7 4809 E0C3 0920 0000 00525000 0
10C8 2200 0000 0000 0000 00526000 0
10C9 4809 C650 0000 0000 00527000 0
10CA 2F5C 0000 0000 0000 00528000 0
10CB 2000 0000 0000 0000 00529000 0
10CC 64EC C0C0 0000 0000 00530000 0
10CD 570C 0000 0000 0000 00531000 0
10CE 4809 0C41 0C10 00F0 00532000 0
10CF 0000 0000 0000 0000 00533000 0
10D0 60E0 0000 0000 0000 00534000 0
10D1 4809 EC40 0030 0000 00535000 0
10D2 4809 EC00 0000 0000 00536000 0
10D3 2200 0000 0000 0000 00537000 0
10D4 4809 0C40 2000 0000 00538000 0
10D5 4809 C650 0000 0000 00539000 0
10D6 2F5C 0000 0000 0000 00540000 0
10D7 2000 0000 0000 0000 00541000 0
10D8 56E0 0000 0000 0000 00542000 0
10D9 5F9C 0000 0000 0000 00543000 0
10DA 4809 C640 0000 0000 00544000 0
10DB 193C 0000 0000 0000 00545000 0
10DC 4809 00C0 0000 0000 00546000 0
10DD 4824 00C3 0000 0000 00547000 0
10DE 194C 00C3 0000 0000 00548000 0
10DF 195C 00C3 0000 0000 00549000 0
10E0 196C 00C3 0000 0000 00550000 0
10E1 197C 00C3 0000 0000 00551000 0
10E2 220C 0000 0000 0000 00552000 0
10E3 4809 0C40 0030 0000 00553000 0
10E4 2F5C 0000 0000 0000 00554000 0
10E5 2000 0000 0000 0000 00555000 0
10E6 56E0 0000 0000 0000 00556000 0
10E7 4809 0C40 0030 0000 00557000 0
10E8 2F5C 0000 0000 0000 00558000 0
10E9 2000 0000 0000 0000 00559000 0
10EA 56E0 0000 0000 0000 00560000 0
10EB 4809 0C40 0030 0000 00561000 0
10EC 238C 00C3 0000 0000 00562000 0
10ED 239C 00C3 0000 0000 00563000 0
10EE 4809 0C40 2000 0000 00564000 0
10EF 4809 E0C3 0920 0000 00565000 0
10F0 0000 0000 0000 0000 00566000 0
10F1 2F5C 0000 0000 0000 00567000 0
10F2 2000 0000 0000 0000 00568000 0
10F3 56E0 0000 0000 0000 00569000 0
10F4 5F9C 0000 0000 0000 00570000 0
10F5 4809 C640 0000 0000 00571000 0
10F6 193C 0000 0000 0000 00572000 0
10F7 4809 00C0 0000 0000 00573000 0
10F8 4824 00C3 0000 0000 00574000 0
10F9 194C 00C3 0000 0000 00575000 0
10FA 195C 00C3 0000 0000 00576000 0
10FB 196C 00C3 0000 0000 00577000 0
10FC 197C 00C3 0000 0000 00578000 0
10FD 220C 0000 0000 0000 00579000 0
10FE 4809 0C40 0030 0000 00580000 0
10FF 56E0 0000 0000 0000 00581000 0

```


1109	233C 0000 0000 0000	CONTENTSIRH - 1 = CPCR	% Y* INTO B	05578C00 0
110A	4809 0C41 0010 0000	B L = BR2		05575000 0
110B	000C 0000 0000 0000	COMP 8 = SAR		05580C00 0
110C	50E0 0000 0000 0000	EMULIN - 1 = CPCR	% (Y*) INTO B	05581C00 0
110D	4809 0C43 0000 0000	B = MIR		05582C00 0
110E	4809 0C00 0800 0000	A3 = B		05583C00 0
110F	2280 0000 0000 0000	CONTENTSRA - 1 = CPCR	% (R(A*)) INTO B	05584C00 0
110G	4809 0C40 2000 0000	B = A2, BHI		05585C00 0
1110	4809 0C00 0000 0000	A2 OR B = MIR, B		05586C00 0
1111	4809 0C5C 0800 0000	EQUIPUT - 1 = CPCR	% SET THE CONDITION BITS	05587C00 0
1112	2F5C 0000 0000 0000	SETCCA - 1 = CPCR		05588C00 0
1113	2000 0000 0000 0000	OPCODE - 1 = MPCR		05589C00 0
1114	56E0 0000 0000 0000			05590C00 0
1115	4809 2C55 0800 0000	LIT AND B = 0		05591C00 0
1116	000C 0000 0000 0000	I5 = LIT		05592C00 0
1117	4809 0C52 0000 0000	0 EOL R		05593C00 0
1118	6810 0000 0000 0000	IF TRUE THEN SKIP	% CHECK FOR H = 0	05594C00 0
1119	2380 0000 0000 0000	CONTENTSIRH - 1 = CPCR	% (R(P*)) INTO B	05595C00 0
111A	4809 0C41 0800 0000	B L = B		05596C00 0
111B	0000 0000 0000 0000	COMP 16 = SAR		05597C00 0
111C	4809 0C5C 1C00 0000	A3 OR B = A3		05598C00 0
111D	633C 0000 0000 0000	IFETCH - 1 = CPCR		05599C00 0
111E	4809 0C00 0000 0000	A3 R = A2	% A3 = (R(P*)) INSTRUCTION	05600C00 0
111F	0CFC 0000 0000 0000	16 = SARI 15 = LIT	% *Y* INTO B	05601C00 0
1120	4809 0C40 0000 0000	A2 + B = MIR	% Y INTO MIR	05602C00 0
1121	4809 0C00 0000 0000	A3 = B		05603C00 0
1122	2280 0000 0000 0000	CONTENTSRA - 1 = CPCR	% (R(A*)) INTO B	05604C00 0
1123	4809 0C40 2000 0000	B = A2, BHI		05605C00 0
1124	4809 0C5C 0800 0000	A2 OR B = MIR, B		05606C00 0
1125	2F5C 0000 0000 0000	EQUIPUT - 1 = CPCR		05607C00 0
1126	2000 0000 0000 0000	SETCCA - 1 = CPCR	% SET THE CONDITION BITS	05608C00 0
1127	56E0 0000 0000 0000	OPCODE - 1 = MPCR		05609C00 0
1128	570C 0000 0000 0000			05610C00 0
1129	4809 0C41 0010 0000	RYMFIELD - 1 = LPCR	% RX TYPE OR	05611C00 0
112A	000C 0000 0000 0000	B L = BR2	% (R(A*)) OR (Y) = (R(A)), SET CC	05612C00 0
112B	60E0 0000 0000 0000	COMP 8 = SAR	% Y INTO B	05613C00 0
112C	4809 0C40 0000 0000	EMULIN - 1 = CPCR	% (Y) INTO B	05614C00 0
112D	4809 0C00 0800 0000	B = MIR		05615C00 0
112E	2280 0000 0000 0000	A3 = B		05616C00 0
112F	4809 0C40 2000 0000	CONTENTSRA - 1 = CPCR	% (R(A*)) INTO B	05617C00 0
1130	4809 0C5C 0800 0000	B = A2, BHI		05618C00 0
1131	2F50 0000 0000 0000	A2 OR B = MIR, B		05619C00 0
1132	2000 0000 0000 0000	EQUIPUT - 1 = CPCR	% SET THE CONDITION BITS	05620C00 0
1133	56E0 0000 0000 0000	SETCCA - 1 = CPCR		05621C00 0
1134	5F9C 0000 0000 0000	OPCODE - 1 = MPCR		05622C00 0
1135	4809 0C40 0010 0000			05623C00 0
1136	190C 0000 0000 0000	OPCODE - 1 = CPCR	% *F* CODE INTO A2	05624C00 0
1137	1809 0000 0000 0000	OPCODE - 1 = MPCR		05625C00 0
1138	482A 0000 0000 0000			05626C00 0
1139	190C 0000 0000 0000			05627C00 0

113A 19A0 00C0 C0C0 C040
1130 19B0 00C0 00C0 C040
113C 19C0 00C0 00C0 C040

OP320:

1130 2280 C0C0 00C0 0060
113E 4809 0C40 00C0 0060
113F 238C 00C0 00C0 0060
1140 0C40 00C0 00C0 0060
1141 4809 CC4C 004C 0060
1142 4809 C0C0 801C 0060
1143 00C0 00C0 00C0 0020
1144 2F5C 00C0 00C0 0060
1145 20C0 00C0 00C0 0060
1146 56E0 00C0 00C0 0040

OP321:

1147 2380 C0C0 00C0 C060
1148 4809 0C40 001C C060
1149 00C0 C0C0 00C0 C060
114A 50E0 C0C0 00C0 C060
114B 4809 CC4C 00C0 C060
114C 4809 E0C0 00C0 C060
114D 228C 00C0 00C0 C060
114E 4809 0C40 00C0 C060
114F 4809 CC4C 00C0 C060
1150 2F5C C0C0 00C0 C060
1151 20C0 00C0 00C0 C060
1152 56E0 00C0 00C0 C040

OP322:

1153 4809 2C55 00C0 0060
1154 00C0 00C0 00C0 0060
1155 4809 0C52 00C0 0060
1156 5819 00C0 00C0 0060
1157 238C 00C0 00C0 C060
1158 48C9 CC4C 00C0 C060
1159 00C0 C0C0 00C0 0020
115A 00C0 00C0 00C0 00C0
115B 5330 00C0 00C0 00C0
115C 4809 E0C0 00C0 C060
115D 00C0 00C0 00C0 C060
115E 48C9 C0C0 00C0 C060
115F 4809 E0C0 00C0 C060
1160 2280 C0C0 00C0 0060
1161 4809 0C40 20C0 C060
1162 4809 CC4C 00C0 C060
1163 2F5C 00C0 00C0 C060
1164 20C0 00C0 00C0 C060
1165 66E0 00C0 C0C0 C040

OP323:

OP321 - 1 = MPCR
OP322 - 1 = MPCR
OP323 - 1 = MPCR

OP320:

CONTENTSRA - 1 = CPCR
B = M1R
CONTENTSRRM - 1 = CPCR
B = A2, BHI
A2 XOR B = M1R, F
A3 R = MAR2
16 = SAR
EQUIPUT - 1 = CPCR
SETCCA - 1 = CPCR
OPCODE - 1 = MPCR

OP321:

CONTENTSRRM - 1 = CPCR
B = M1R
COMPL - 1 = CPCR
FHLIN - 1 = CPCR
B = M1R
A3 = B
CONTENTSRA - 1 = CPCR
B = A2, BHI
A2 XOR B = M1R, F
EQUIPUT - 1 = CPCR
SETCCA - 1 = CPCR
OPCODE - 1 = MPCR

OP322:

LIT AND B = B
15 = LIT
0 EOL B
IF TRUE THEN SKIP
CONTENTSRRM - 1 = CPCR
B L = B
COMPL - 1 = CPCR
A3 OR B = A3
FHLIN - 1 = CPCR
12 R SAR, 15 = LIT
A2 + B = M1R
A3 = B
CONTENTSRA - 1 = CPCR
B = A2, BHI
A2 XOR B = M1R, F
EQUIPUT - 1 = CPCR
SETCCA - 1 = CPCR
OPCODE - 1 = MPCR

OP323:

05639C00 0
05640C00 0
05641C00 0
05642C00 0
05643C00 0
05644C00 0
05645C00 0
05646C00 0
05647C00 0
05648C00 0
05649C00 0
05650C00 0
05651C00 0
05652C00 0
05653C00 0
05654C00 0
05655C00 0
05656C00 0
05657C00 0
05658C00 0
05659C00 0
05660C00 0
05661C00 0
05662C00 0
05663C00 0
05664C00 0
05665C00 0
05666C00 0
05667C00 0
05668C00 0
05669C00 0
05670C00 0
05671C00 0
05672C00 0
05673C00 0
05674C00 0
05675C00 0
05676C00 0
05677C00 0
05678C00 0
05679C00 0
05680C00 0
05681C00 0
05682C00 0
05683C00 0
05684C00 0
05685C00 0
05686C00 0
05687C00 0
05688C00 0
05689C00 0
05690C00 0
05691C00 0
05692C00 0
05693C00 0
05694C00 0
05695C00 0
05696C00 0
05697C00 0

1198	51CC	CC00	CC0C	0060	REGSTACK - 1 = CPCR	X R(A) INTO MAR2	05758000 D
1199	4809	0F41	0070	00FC	BHAR L = BR1	X TEMP STORAGE	05759000 D
1198	000C	00C0	0000	0030	COMP 0 = SAR		05760000 D
1199	4809	0F45	003C	00FC	BHAR + 1 = R	X R(A+1)	05761000 D
1190	51CC	00C0	003C	0060	REGSTACK - 1 = CPCR	X R(A+1) INTO PAR2	05762000 D
1191	2F3C	0000	00C0	006C	INPUT - 1 = CPCR	X R(A+1) INTO B	05763000 D
1192	4809	0C40	0030	00FC	0 = A2, BHI	X (R(A+1)) AND (Y)	05764000 D
1193	4809	CC56	0030	00FC	ASR	REFERENCE BR1	05765000 D
1194	4809	0C43	001C	00FC	BHAR R = MAR2	X R(A)	05766000 D
1195	000C	00C0	0000	001C	0 = SAR	X (R(A)) INTO B	05767000 D
1196	2F30	00C0	0000	0060	INPUT - 1 = CPCR		05768000 D
1197	4809	0C40	0020	00FC	P = A2, BHI		05769000 D
1198	4809	CC5C	0030	00FC	A2 OR B = MIR, B		05770000 D
1199	2F50	C000	007C	006C	EUPTUT - 1 = CPCR		05771000 D
11A0	200C	00C0	00C0	0060	SETPCA - 1 = CPCR		05772000 D
11A1	56EC	00C0	00C0	0040	OPCODE - 1 = MPCR	X SET THE CONDITION BITS	05773000 D
11A2	4809	0C43	001C	00FC			05774000 D
11A3	000C	00C0	0000	001C			05775000 D
11A4	2F30	00C0	0000	0060			05776000 D
11A5	4809	0C40	0020	00FC			05777000 D
11A6	4809	CC5C	0030	00FC			05778000 D
11A7	2F50	C000	007C	006C			05779000 D
11A8	200C	00C0	00C0	0060			05780000 D
11A9	56EC	00C0	00C0	0040			05781000 D
11AA	4809	2C56	003C	00FC			05782000 D
11AB	00FC	CC00	CC00	00E0			05783000 D
11AC	4809	0C52	0070	00FC			05784000 D
11AD	6819	0070	00FC	00FC			05785000 D
11AE	2380	007C	009C	0060			05786000 D
11AF	4809	0C41	0030	00FC			05787000 D
11B0	000C	00C0	0000	002C			05788000 D
11B1	4809	E5C5	1000	00FC			05789000 D
11B2	533C	00C0	00C0	0060			05790000 D
11B3	000C	E5C3	AC00	00FC			05791000 D
11B4	000C	CC40	00C0	0020			05792000 D
11B5	4809	CC40	00C0	0020			05793000 D
11B6	4809	E0C0	8B00	00FC			05794000 D
11B7	80FC	00C0	0070	00FC			05795000 D
11B8	4809	2C56	0030	00FC			05796000 D
11B9	51CC	0000	0070	0060			05797000 D
11BA	4809	0F41	002C	0030			05798000 D
11BB	00CC	00C0	0070	0030			05799000 D
11BC	4809	0F46	003C	006C			05800000 D
11BD	51CC	00C0	0030	006C			05801000 D
11BE	2F30	00C0	0020	0060			05802000 D
11BF	4809	CC43	2000	00FC			05803000 D
11C0	4809	CC56	003C	00FC			05804000 D
11C1	4809	CC00	0070	00FC			05805000 D
11C2	4809	0F43	901C	00FC			05806000 D
11C3	0000	00C0	003C	0010			05807000 D
11C4	2F30	CC00	0070	0060			05808000 D
11C5	4809	0C40	2000	00FC			05809000 D
11C6	4809	CC5C	0030	00FC			05810000 D
11C7	2F50	CC00	0000	0060			05811000 D
11C8	200C	00C0	0000	0060			05812000 D
11C9	56EC	00C0	00C0	0040			05813000 D
11CA	570C	CC00	CC0C	0060			05814000 D
11CB	4809	CC41	CC00	00FC			05815000 D
11CC	0000	00C0	0000	0030			05816000 D

OP332:

OP333:

230C	00D0	0070	0060	0050	0040	0030	0020	0010	0000	CONTENSTRM - 1 = CPCR	% Y* INTO B	05878C00
4809	0C41	0010	0010	00F0						B L = BR2		05879C00
500C	1C00	0010	0030	0030						COMP 8 = SAR	% (Y*) INTO B	05880C00
500C	1C00	0030	0030	0060						EMULIN - 1 = CPCR	% (Y*) INTO B	05881C00
500C	0C40	0030	00F0							R = MIR	% TEMP STORAGE	05882C00
4809	0C40	0030	00F0							A3 R = B		05883C00
4809	0C40	0080	00F0							% = SAR1 15 = LIT		05884C00
80FC	00C0	0000	00F0							LIT AND B = B	% ISOLATE *A*	05885C00
4809	0C55	0080	00F0							REGSTACK - 1 = CPCR		05886C00
51CC	0000	0000	0060							EMPTUT - 1 = CPCR	% (R(A)) INTO E	05887C00
2F50	0000	0000	0060							B = A2		05888C00
4809	0C40	0030	00F0							BMAR + 1 = B	% R(A+1)	05889C00
4809	0C45	0030	00F0							REGSTACK - 1 = CPCR	% R(A+1) INTO B	05890C00
2F5C	0000	0000	0060							EMPTUT - 1 = CPCR	% R(A+1) INTO B	05891C00
4809	0C55	0030	00F0							A2 AND B = A2	% (R(A)) AND (R(A+1))	05892C00
4809	0C40	1030	00F0							R = A3, PH1		05893C00
4809	0C58	0010	00F0							A3 AND B = 0	% (Y*) AND (R(A+1))	05894C00
200C	0C00	0030	00F0							SE1CC - 1 = CPCR	% SET THE CONDITION BITS	05895C00
80E0	0000	0030	0060							OPCODE - 1 = MPCH		05896C00
4809	0C56	0030	00F0							LIT AND B = B	% TYPE RK COMPARE MASKED (CONSTANT)	05897C00
4809	0C52	0030	00F0							15 = LIT	% (R(A)) AND (R(A+1)) Y AND (R(A+1))	05898C00
6819	0000	0030	00F0							IF TRUE THEN SKIF	% AND SET CONDITION BITS	05899C00
230C	0000	0030	0060							CONTENSTRM - 1 = CPCR	% ISOLATE M	05900C00
4809	0C41	0030	00F0							B L = B	% CHECK FOR M = 0	05901C00
4809	0C5C	1000	00F0							COMP 16 = SAR	% (R(H)) INTO B	05902C00
4809	0C5C	1000	00F0							A3 OR B = A3	% A3 = (R(M))/INSTRUCTION	05903C00
5330	0000	0000	0060							IFETCH - 1 = CPCR	% Y* INTO B	05904C00
4809	0C03	0030	00F0							A3 R = A2	% (R(M)) INTO A2	05911C00
4809	0C03	0030	00F0							16 = SAR	% Y INTO D	05912C00
4809	0C04	0030	00F0							A2 + B = MIR		05913C00
4809	0C04	0030	00F0							A3 R = B	% Y INTO D	05914C00
50FC	0000	0030	0080							LIT AND B = B	% ISOLATE *A*	05915C00
4809	0C55	0030	00F0							REGSTACK - 1 = CPCR	% (R(A)) INTO B	05916C00
51CC	0000	0000	0060							EMPTUT - 1 = CPCR	% (R(A)) INTO E	05917C00
4809	0C40	0030	00F0							B = A2		05918C00
4809	0C40	0030	00F0							BMAR + 1 = B	% R(A+1)	05919C00
4809	0C45	0030	00F0							REGSTACK - 1 = CPCR	% R(A+1) INTO B	05920C00
50FC	0000	0030	0080							EMPTUT - 1 = CPCR	% R(A+1) INTO B	05921C00
2F5C	0000	0000	0060							A2 AND B = A2	% (R(A)) AND (R(A+1))	05922C00
4809	0C52	0030	00F0							R = A3, PH1	% Y AND (R(A+1))	05923C00
4809	0C58	0010	00F0		</							

LINE	NO	OPERATION	DATA
1229	570C	0000	0030 006C
1230	570C	0000	0030 006C
1231	4809	0C41	0011 00FC
1232	4809	0C41	0011 00FC
1233	0C00	0FC0	0C00 0C30
1234	0C00	0FC0	0C00 0C30
1235	50E0	0120	0000 006C
1236	50E0	0120	0000 006C
1237	4809	0C43	0030 00FC
1238	4809	0C43	0030 00FC

[illegible]

125E 3000 0000 0000 0000
125F 3000 0000 0000 0000
1260 3000 0000 0000 0000
1261 3000 0000 0000 0000
1262 3000 0000 0000 0000
1263 3000 0000 0000 0000
1264 3000 0000 0000 0000
1265 3000 0000 0000 0000
1266 3000 0000 0000 0000
1267 3000 0000 0000 0000
1268 3000 0000 0000 0000
1269 3000 0000 0000 0000
1270 3000 0000 0000 0000
1271 3000 0000 0000 0000
1272 3000 0000 0000 0000
1273 3000 0000 0000 0000
1274 3000 0000 0000 0000
1275 3000 0000 0000 0000
1276 3000 0000 0000 0000
1277 3000 0000 0000 0000
1278 3000 0000 0000 0000
1279 3000 0000 0000 0000
1280 3000 0000 0000 0000
1281 3000 0000 0000 0000
1282 3000 0000 0000 0000
1283 3000 0000 0000 0000
1284 3000 0000 0000 0000
1285 3000 0000 0000 0000
1286 3000 0000 0000 0000
1287 3000 0000 0000 0000
1288 3000 0000 0000 0000
1289 3000 0000 0000 0000
1290 3000 0000 0000 0000
1291 3000 0000 0000 0000

COMP 8 = SAR
EMULIN - 1 = CPCH
B L = B, A3
COMP 16 = SAR
IF NOT MST THEN STEP ELSE SKIP
P351 - 1 = MPCR
SET11 - 1 = MPCR
I L = B
COMP 30 = SAR
A3 OR B = MIR, D
16 = SAR
RHAR L = BR2
COMP 8 = SAR
EMULOUT - 1 = CPCH
OPCODE - 1 = MPCR
A1 C = A1, CSAR
25 = SAR
A1 AND B110 C = A1
B C = B
30 = SAR
LIT OR B C = MIR
J = LIT, B = SAR
RHAR L = BR2
COMP 8 = SAR
EMULOUT - 1 = CPCH
OPCODE - 1 = MPCR
P351:
A1 AND B110 C = A1
B C = B
30 = SAR
LIT OR B C = MIR
J = LIT, B = SAR
RHAR L = BR2
COMP 8 = SAR
EMULOUT - 1 = CPCH
OPCODE - 1 = MPCR
P352:
LIT AND P = B
15 = LIT
O EOL B
IF TRUE THEN SKIP
CONTENTSRN - 1 = CPCH
P L = B
COMP 16 = SAR
A3 OR B = A3
IFETCH - 1 = CPCH
A3 R = A2
PSW = LIT, 16 = SAR
A2 + B = MIR
LIT = HAR2
EMULOUT - 1 = CPCH
A1 OR B10 = A1
OPCODE - 1 = MPCR
P353:
RHARFIELD - 1 = CPCH
B L = BR2, A3
COMP 8 = SAR
EMULIN - 1 = CPCH
B L = B
COMP 16 = SAR
IF NOT MST THEN STEP ELSE SKIP
P353 - 1 = MPCR

05598100 0
05599100 0
06000100 0
06001100 0
06002100 0
06003100 0
06004100 0
06005100 0
06006100 0
06007100 0
06008100 0
06009100 0
06010100 0
06011100 0
06012100 0
06013100 0
06014100 0
06015100 0
06016100 0
06017100 0
06018100 0
06019100 0
06020100 0
06021100 0
06022100 0
06023100 0
06024100 0
06025100 0
06026100 0
06027100 0
06028100 0
06029100 0
06030100 0
06031100 0
06032100 0
06033100 0
06034100 0
06035100 0
06036100 0
06037100 0
06038100 0
06039100 0
06040100 0
06041100 0
06042100 0
06043100 0
06044100 0
06045100 0
06046100 0
06047100 0
06048100 0
06049100 0
06050100 0
06051100 0
06052100 0
06053100 0
06054100 0
06055100 0
06056100 0
06057100 0

X (Y*) INTO R
X (Y*) INTO HS WORD OF B, A3
X B TO THE ADDR
ELSE SKIP
X POSITIVE CASI
X SET THE CONDITION R115
X SET EIT 14 OF (Y*)
X RESTORE Y* INTO PR2
X PUT (C BIT 9 INTO LS BIT
X SET PIT 9 TO 0, AND RESTORE A1
X SET (Y*) BITS 14,15
X PESTORE Y* INTO BR2
X TYPE RK REMOTE EXECUTE
X EXECUTE (Y*) (P) + 2 = P
X ISOLATE Y*


```

1702 4809 0C52 0000 00F0 0 00L B
1703 63C9 0000 0000 00F0 IF FALSE THEN SET LCI
1704 18A0 0000 0000 0000 OP40X - 1 = MPCR

OP40X01: CHECKCC - 1 = CPCR
0 00L B
IF FALSE THEN SET LCI
OP40X - 1 = MPCR

OP40X02: CHECKCC - 1 = CPCR
LIT GE0 B
2 = LIT
IF FALSE THEN SET LCI
OP40X - 1 = MPCR

OP40X03: CHECKCC - 1 = CPCR
LIT EOL E
3 = LIT
IF FALSE THEN SET LCI
OP40X - 1 = MPCR

OP40X04: A1 R = A2
27 = SAR
A2
IF NOT LST THEN SET LCI
OP40X - 1 = MPCR

OP40X05: A1 R = A2
28 = SAR
A2
IF NOT LST THEN SET LCI
OP40X - 1 = MPCR

OP40X10: OP40X - 1 = MPCR

OP40X11: WAIT
STEP
OP40X - 1 = MPCR

OP40X12: A1 R = A2
29 = SAR
A2
IF NOT LST THEN SET LCI ELSE SKIP
OP40X - 1 = MPCR
WAIT
STEP
OP40X - 1 = MPCR

OP40X13: A1 R = A2
30 = SAR
A2

```

06118C00 0
 06119C00 0
 06120C00 0
 06121C00 0
 06122C00 0
 06123C00 0
 06124C00 0
 06125C00 0
 06126C00 0
 06127C00 0
 06128C00 0
 06129C00 0
 06130C00 0
 06131C00 0
 06132C00 0
 06133C00 0
 06134C00 0
 06135C00 0
 06136C00 0
 06137C00 0
 06138C00 0
 06139C00 0
 06140C00 0
 06141C00 0
 06142C00 0
 06143C00 0
 06144C00 0
 06145C00 0
 06146C00 0
 06147C00 0
 06148C00 0
 06149C00 0
 06150C00 0
 06151C00 0
 06152C00 0
 06153C00 0
 06154C00 0
 06155C00 0
 06156C00 0
 06157C00 0
 06158C00 0
 06159C00 0
 06160C00 0
 06161C00 0
 06162C00 0
 06163C00 0
 06164C00 0
 06165C00 0
 06166C00 0
 06167C00 0
 06168C00 0
 06169C00 0
 06170C00 0
 06171C00 0
 06172C00 0
 06173C00 0
 06174C00 0
 06175C00 0
 06176C00 0
 06177C00 0

* CC NOT EQUAL OR NOT 0, JUMP NOT EQUAL
 * CC >= OR + J JUMP GTR OR EQUAL
 * CC < OR - J JUMP LESS
 * IF OVERFLOW SET, JUMP, JUMP OVERFLOW
 * IF CARRY, JUMP CARRY
 * JUMP AFTER STOP
 * MACHINE STOP
 * IF KEY 1 SET, STOP, JUMP.
 * IF KEY 2 SET, STOP, JUMP


```

131E 0000 0000 0000 0020      16 = SAR
131F 4809 AC14 407C 60F0      A1 L = A1
1320 4809 AC5C 407C 60F0      A1 OR D = A1
1321 66E0 0070 0070 0040      OPCODE - 1 = MPCR

OP401:
1322 559C 0070 0070 004C      R11 - 1 = MPCR

OPCODE41:
1323 5F90 00C0 0030 0060      YECODE - 1 = CPCR
1324 4809 C640 007C C0FC      A2 + MPCR = AMPCR
1325 1CAC 0003 C030 00C0      OP41F - 1 = AMPCR
1326 4009 00C0 C00C 00F0      STEP
1327 4824 0063 0070 00FC      EXEC

1328 1CB0 00C0 007C 0040      OP41F:
1329 1CC0 0070 C07C 0040      OP411 - 1 = MPCR
132A 00C0 007C 004C      OP412 - 1 = MPCR
132B 1CEC 0060 000C 0040      OP413 - 1 = MPCR

OP410:
132C 2260 00C0 0070 C060      CONTENTSRA - 1 = CPCR
132D 4809 0052 C030 0060      D EOL 0
132E 56E0 00C0 007C 00F0      IF TRUE THEN STEP ELSE SKIP
132F 56E0 007C 0070 0040      OPCODE - 1 = MPCR
1330 4809 0040 207C 00F0      R = A2
1331 4809 0070 207C 00F0      A2 - 1 = M1R
1332 2E58 C030 00C0 0060      OUTPUT - 1 = CPCR
1333 2380 00C0 C00C 0060      CONTENTSRA - 1 = CPCR
1334 1809 AC14 407C 60F0      A1 R = A1
1335 0000 00C0 007C C020      16 = SAR
1336 1809 AC14 407C 60F0      A1 OR D = A1
1337 1809 AC5C 407C 60F0      A1 OR D = A1
1338 56E0 00C0 0000 0040      OPCODE - 1 = MPCR

OP411:
1339 2260 00C0 007C 0060      CHECKCC - 1 = CPCR
133A 1809 C0C2 007C C0FC      R EOL 0
133B 680B 00C0 007C 60F0      IF TRUE THEN STEP ELSE SKIP
133C 56E0 00C0 007C 0040      OPCODE - 1 = MPCR
133D 5590 0070 0070 C04C      R11 - 1 = MPCR

OP412:
133E 2260 00C0 C030 0060      PK TYPE INDEX JUMP
133F 1809 0052 C030 0060      IF (5(A)) NE 0, (R(A)) - 1 INTO R(A),
1340 680B 00C0 007C C0FC      IF EOL 0
1341 56E0 00C0 007C C04C      IF TRUE THEN STEP ELSE SKIP
1342 4809 0040 207C 00FC      OPCODE - 1 = MPCR

```



```

372 4809 0C40 8000 C0F0      % (P) + 1 INTO R(A) (RCHS) INTO P
373 480F 0C40 8000 C0E0      % B CONTAINS "A"
374 4809 2C56 0830 008C      REGSTACK - 1 = CPCCR
375 31C0 00C3 0030 C0FC      % ADDRESS OF R(A) IN MAR2
376 4809 A001 2C00 00F0      A1 L = A2
377 30C0 00C0 0000 0020      16 = SAR
378 4809 C0C0 A000 0020      A2 R = A2
379 4009 0C40 0030 0010      A2 + 1 = MIR
380 31C0 00C3 0030 0060      EOUTPUT - 1 = CPCCR
381 4809 0C40 0030 0060      CONTENTSRM - 1 = CPCCR
382 4809 0C40 0030 0060      COMP 16 = SAR
383 4809 0C40 0030 0060      COMP 16 = SAR
384 4809 0C40 0030 0060      A1 L = A2
385 4809 0C40 0030 0060      A1 OR B = A1
386 4809 0C40 0030 0060      OPCODE - 1 = HPCR
387 4809 0C40 0030 0060      B R = B
388 4809 0C40 0030 0060      q = SARA 15 = LIT
389 4809 0C40 0030 0060      LIT AND B = F
390 4809 0C40 0030 0060      REGSTACK - 1 = CPCCR
391 4809 0C40 0030 0060      A1 L = A2
392 4809 0C40 0030 0060      16 = SARI 2 = LIT
393 4809 0C40 0030 0060      A2 R = A2
394 4809 0C40 0030 0060      A2 + LIT = MIR
395 4809 0C40 0030 0060      EOUTPUT - 1 = CPCCR
396 4809 0C40 0030 0060      A3 = B
397 4809 0C40 0030 0060      RYHFIELD - 1 = CFCH
398 4809 0C40 0030 0060      P L = BR2
399 4809 0C40 0030 0060      COMP 8 = SAR
400 4809 0C40 0030 0060      EYULIN - 1 = CPCCR
401 4809 0C40 0030 0060      B = A3
402 4809 0C40 0030 0060      IFETCH - 1 = CPCCR
403 4809 0C40 0030 0060      A3 + B = B
404 4809 0C40 0030 0060      A1 R = A1
405 4809 0C40 0030 0060      COMP 16 = SAR
406 4809 0C40 0030 0060      A1 L = A1
407 4809 0C40 0030 0060      A1 OR D = A1
408 4809 0C40 0030 0060      OPCODE - 1 = HPCR
409 4809 0C40 0030 0060      B R = B
410 4809 0C40 0030 0060      q = SARA 15 = LIT
411 4809 0C40 0030 0060      LIT AND B = F
412 4809 0C40 0030 0060      REGSTACK - 1 = CPCCR
413 4809 0C40 0030 0060      A1 L = A2
414 4809 0C40 0030 0060      16 = SARI 2 = LIT
415 4809 0C40 0030 0060      A2 R = A2
416 4809 0C40 0030 0060      A2 + LIT = MIR
417 4809 0C40 0030 0060      EOUTPUT - 1 = CPCCR
418 4809 0C40 0030 0060      A3 = B
419 4809 0C40 0030 0060      RYHFIELD - 1 = CFCH
420 4809 0C40 0030 0060      P L = BR2
421 4809 0C40 0030 0060      COMP 8 = SAR
422 4809 0C40 0030 0060      EYULIN - 1 = CPCCR
423 4809 0C40 0030 0060      B = A3
424 4809 0C40 0030 0060      IFETCH - 1 = CPCCR
425 4809 0C40 0030 0060      A3 + B = B
426 4809 0C40 0030 0060      A1 R = A1
427 4809 0C40 0030 0060      COMP 16 = SAR
428 4809 0C40 0030 0060      A1 L = A1
429 4809 0C40 0030 0060      A1 OR D = A1
430 4809 0C40 0030 0060      OPCODE - 1 = HPCR
431 4809 0C40 0030 0060      B R = B
432 4809 0C40 0030 0060      q = SARA 15 = LIT
433 4809 0C40 0030 0060      LIT AND B = F
434 4809 0C40 0030 0060      REGSTACK - 1 = CPCCR
435 4809 0C40 0030 0060      A1 L = A2
436 4809 0C40 0030 0060      16 = SARI 2 = LIT
437 4809 0C40 0030 0060      A2 R = A2
438 4809 0C40 0030 0060      A2 + LIT = MIR
439 4809 0C40 0030 0060      EOUTPUT - 1 = CPCCR
440 4809 0C40 0030 0060      A3 = B
441 4809 0C40 0030 0060      RYHFIELD - 1 = CFCH
442 4809 0C40 0030 0060      P L = BR2
443 4809 0C40 0030 0060      COMP 8 = SAR
444 4809 0C40 0030 0060      EYULIN - 1 = CPCCR
445 4809 0C40 0030 0060      B = A3
446 4809 0C40 0030 0060      IFETCH - 1 = CPCCR
447 4809 0C40 0030 0060      A3 + B = B
448 4809 0C40 0030 0060      A1 R = A1
449 4809 0C40 0030 0060      COMP 16 = SAR
450 4809 0C40 0030 0060      A1 L = A1
451 4809 0C40 0030 0060      A1 OR D = A1
452 4809 0C40 0030 0060      OPCODE - 1 = HPCR
453 4809 0C40 0030 0060      B R = B
454 4809 0C40 0030 0060      q = SARA 15 = LIT
455 4809 0C40 0030 0060      LIT AND B = F
456 4809 0C40 0030 0060      REGSTACK - 1 = CPCCR
457 4809 0C40 0030 0060      A1 L = A2
458 4809 0C40 0030 0060      16 = SARI 2 = LIT
459 4809 0C40 0030 0060      A2 R = A2
460 4809 0C40 0030 0060      A2 + LIT = MIR
461 4809 0C40 0030 0060      EOUTPUT - 1 = CPCCR
462 4809 0C40 0030 0060      A3 = B
463 4809 0C40 0030 0060      RYHFIELD - 1 = CFCH
464 4809 0C40 0030 0060      P L = BR2
465 4809 0C40 0030 0060      COMP 8 = SAR
466 4809 0C40 0030 0060      EYULIN - 1 = CPCCR
467 4809 0C40 0030 0060      B = A3
468 4809 0C40 0030 0060      IFETCH - 1 = CPCCR
469 4809 0C40 0030 0060      A3 + B = B
470 4809 0C40 0030 0060      A1 R = A1
471 4809 0C40 0030 0060      COMP 16 = SAR
472 4809 0C40 0030 0060      A1 L = A1
473 4809 0C40 0030 0060      A1 OR D = A1
474 4809 0C40 0030 0060      OPCODE - 1 = HPCR
475 4809 0C40 0030 0060      B R = B
476 4809 0C40 0030 0060      q = SARA 15 = LIT
477 4809 0C40 0030 0060      LIT AND B = F
478 4809 0C40 0030 0060      REGSTACK - 1 = CPCCR
479 4809 0C40 0030 0060      A1 L = A2
480 4809 0C40 0030 0060      16 = SARI 2 = LIT
481 4809 0C40 0030 0060      A2 R = A2
482 4809 0C40 0030 0060      A2 + LIT = MIR
483 4809 0C40 0030 0060      EOUTPUT - 1 = CPCCR
484 4809 0C40 0030 0060      A3 = B
485 4809 0C40 0030 0060      RYHFIELD - 1 = CFCH
486 4809 0C40 0030 0060      P L = BR2
487 4809 0C40 0030 0060      COMP 8 = SAR
488 4809 0C40 0030 0060      EYULIN - 1 = CPCCR
489 4809 0C40 0030 0060      B = A3
490 4809 0C40 0030 0060      IFETCH - 1 = CPCCR
491 4809 0C40 0030 0060      A3 + B = B
492 4809 0C40 0030 0060      A1 R = A1
493 4809 0C40 0030 0060      COMP 16 = SAR
494 4809 0C40 0030 0060      A1 L = A1
495 4809 0C40 0030 0060      A1 OR D = A1
496 4809 0C40 0030 0060      OPCODE - 1 = HPCR
497 4809 0C40 0030 0060      B R = B
498 4809 0C40 0030 0060      q = SARA 15 = LIT
499 4809 0C40 0030 0060      LIT AND B = F
500
```



```

13A5 4809 AC00 CC00 00F0 06418000 0
13A6 30C0 00C3 0000 0020 06419000 0
13A7 4809 A0C1 4000 00F0 06420000 0
13A8 4809 AC5C 4000 00F0 06421000 0
13A9 66EC 0003 0000 0040 06422000 0
13AA 5F00 00C0 00C0 0060 06423000 0
13AB 4809 C640 0040 00F0 06424000 0
13AC 103C 0003 0000 00C0 06425000 0
13AD 4809 00C0 00C0 00F0 06426000 0
13AE 482A 00C0 00C0 00F0 06427000 0
13AF 30F0 0000 0070 0040 06428000 0
13B0 1040 00C3 0000 0040 06429000 0
13B1 105C 00C0 00C0 0040 06430000 0
13B2 1060 00C0 0000 0040 06431000 0
13B3 226C 00C0 CC00 0060 06432000 0
13B4 4809 0C52 0070 00F0 06433000 0
13B5 6819 0000 0070 00F0 06434000 0
13B6 66E0 00C0 0000 0040 06435000 0
13B7 2809 A0C1 2000 00F0 06436000 0
13B8 00CC 0003 0070 0020 06437000 0
13B9 4809 C0C0 A000 00F0 06438000 0
13BA 4809 C0C0 0000 00F0 06439000 0
13BB 4809 C0C1 2000 00F0 06440000 0
13BC 4809 E0C0 6000 00F0 06441000 0
13BD 4809 C0C1 0000 00F0 06442000 0
13BE 4809 00C0 0000 00F0 06443000 0
13BF 4809 00C0 0000 00F0 06444000 0
13C0 4809 00C0 0000 00F0 06445000 0
13C1 4809 00C0 0000 00F0 06446000 0
13C2 4809 2557 0000 00F0 06447000 0
13C3 07EC 0000 0070 0040 06448000 0
13C4 4809 C000 0070 00F0 06449000 0
13C5 0000 00C3 0070 0040 06450000 0
13C6 2C19 C0FE 9010 00F0 06451000 0
13C7 4809 CC40 9C10 00F0 06452000 0
13C8 61E0 00C0 0000 0060 06453000 0
13C9 4809 E0C0 8000 00F0 06454000 0
13CA 00C0 00C0 0000 0010 06455000 0
13CB 4809 CC46 C0C0 00F0 06456000 0
13CC 4809 A0C0 C000 00F0 06457000 0
13CD 0000 0000 0070 0020 06458000 0
13CE 4809 AC01 4000 00F0 06459000 0
13CF 4809 AC5C 4000 00F0 06460000 0
13D0 66EC 0000 0000 0040 06461000 0
13D1 4809 2C55 0000 00F0 06462000 0
13D2 00FF 00C0 0000 00E0 06463000 0
13D3 4809 0C52 0030 00F0 06464000 0

```

```

* CLEAR THE OLD PAR
* PREPARE TO RECEIVE NEW PAR
* CREATE NEW PAR
* JUMP, LINK MEMORY
* RETURNED IN A2
* RI TYPE 1, LOCAL JUMP, LINK MEMORY
* ONLY EXECUTE IF CC = 0
* CHECK THE CC CODE
* CHECK IF CC = 0
* PAR INTO A2 (MS WORD)
* PAR INTO A2 (LS WORD)
* PAR INTO UNW
* RESTORE INSTRUCTION INTO B
* SIGN BIT IN PS BIT OF B
* SET LCI
* IF "D" MAGNITUDE IN LSB OF E
* "D" MAGNITUDE IN B
* "D" MAGNITUDE IN B
* IF LCI THEN A2 - B = BR2,A3; SKIP
* A2 + B = BR2,A3
* (P) + D
* (P) + 1 INTO (P) + D
* (P) + D + 1 INTO B
* CLEAR OLD PAR
* (P) + 2 INTO (P) + 1 INTO P
* ISolate "N" FIELD
* IS "H" = 0 ?

```

```

OP00C43:
XFCODE - 1 = CPCR
A2 + AMPCR = AMPCR
OP43F - 1 = AMPCR
STEP
EXEC
FAULT - 1 = FPCR
OP431 - 1 = FPCR
OP432 - 1 = FPCR
OP433 - 1 = MPCR

OP431:
CHECKCC - 1 = CPCR
0 EOL B
IF TRUE THEN SKIP
OPCODE - 1 = MPCR
A1 L = A2; IF LCI
COMP 16 = SAR
A2 R = A2
A2 + 1 = HIR
A2 L = A2
A3 = B
B L = C,CSAR
COMP 24 = SAR
IF INST THEN - B = E;
IF "D" MAGNITUDE IN LSB OF E
LIT AND B L = 0
127 = LIT; COMP 16 = SAR
STEP
B = SAR
IF LCI THEN A2 - B = BR2,A3; SKIP
A2 + B = BR2,A3
CMULOUT - 1 = CPCR
A3 R = B
B = SAR
B + 1 = B
A1 R = A1
COMP 16 = SAR
A1 L = A1
OPCODE - 1 = MPCR

OP432:
LIT AND B = E
15 = LIT
B EOL D

```



```

1402 5619 0C30 0C30 00F0
IF TRUE THEN SKIP
OPCODE - 1 = MPCR
CONTENISRA - 1 = CPCR
% RETURNS (R(H)) IN B
% CLEAR OLD PAR
AL R = A1
COMP 16 = SAR
AL L = A1
AL R = A1
OPCODE - 1 = MPCR
% CREATE NEW PAR
%
%
% RI TYPE 1 LOCAL JUMP EQUAL
% IF (CC) = 0R 0, (P) + D INTO P
% RETURNS CC BITS IN B
B EOL 0
CHECKCC - 1 = CPCR
B EOL 0
IF TRUE THEN SKIP
OPCODE - 1 = MPCR
R11 - 1 = MPCR
% (P) + D = P
%
%
% RK TYPE JUMP ZERO
% IF (R(A)) 0, Y INTO P
% GET (R(A)) INB
CONTENISRA - 1 = CPCR
% RETURNS (R(H)) IN B
B EOL 0
IF TRUE THEN SKIP
EUMP - 1 = MPCR
A3 AND LIT = A3
15 = LIT
A3 EOL 0
IF TRUE THEN 0 = B1 SKIP
CONTENISRA - 1 = CPCR
% RETURN (R(H)) IN B
% Y INTO B
% PAR VALUE IN LS WORD OF E
% CLEAR OLD PAR
AL L = A1
AL R = A1
COMP 16 = SAR
AL L = A1
AL R = A1
OPCODE - 1 = MPCR
% CREATE NEW PAR
%
%
% RX TYPE JUMP ZERO
% IF (R(A)) = C, (Y) INTO P
% (R(A)) INTO B
CONTENISRA - 1 = CPCR
% RETURNS (R(H)) IN B
B EOL 0
IF TRUE THEN SKIP
PUMP - 1 = MPCR
A3 = 9
RXFIELD - 1 = CPCR
B L = BR2
COMP 8 = SAR
COMP 8 = SAR
EULIN - 1 = CPCR
AL R = A1
COMP 16 = SAR
AL L = A1
AL R = A1
OPCODE - 1 = MPCR
% CREATE NEW PAR
%
%
% JUMP NOT ZERO
% RETURN *f* FIELD IN LSD OF A2
XFCODE - 1 = CPCR

```



```

1439 4809 2C56 0670 00F0 06558E00 0
1440 51C0 0070 0070 0060 06659100 0
1441 2F30 0070 0070 0060 06660100 0
1442 4809 0C52 0000 00F0 06661100 0
1443 4809 0052 0000 00F0 06662100 0
1444 5000 0070 0070 00F0 06663000 0
1445 403C 0070 0070 0060 06664000 0
1446 57D0 0070 0070 0060 06665100 0
1447 4809 0C41 0010 00F0 06666100 0
1448 4809 0C41 0010 00F0 06667100 0
1449 5000 0070 0070 0060 06668100 0
1450 4809 0C41 003C 00F0 06669100 0
1451 4809 0C41 003C 00F0 06670100 0
1452 5000 0070 0070 0060 06671000 0
1453 4809 0C41 003C 00F0 06672100 0
1454 4809 0C41 003C 00F0 06673100 0
1455 5000 0070 0070 0060 06674100 0
1456 4809 0C41 003C 00F0 06675100 0
1457 4809 0C41 003C 00F0 06676100 0
1458 5000 0070 0070 0060 06677100 0
1459 4809 0C41 003C 00F0 06678100 0
1460 4809 0C41 003C 00F0 06679100 0
1461 5000 0070 0070 0060 06680100 0
1462 4809 0C41 003C 00F0 06681100 0
1463 4809 0C41 003C 00F0 06682100 0
1464 5000 0070 0070 0060 06683100 0
1465 4809 0C41 003C 00F0 06684100 0
1466 4809 0C41 003C 00F0 06685100 0
1467 5000 0070 0070 0060 06686100 0
1468 4809 0C41 003C 00F0 06687100 0
1469 4809 0C41 003C 00F0 06688100 0
1470 5000 0070 0070 0060 06689100 0
1471 4809 0C41 003C 00F0 06690100 0
1472 4809 0C41 003C 00F0 06691100 0
1473 5000 0070 0070 0060 06692100 0
1474 4809 0C41 003C 00F0 06693100 0
1475 4809 0C41 003C 00F0 06694100 0
1476 5000 0070 0070 0060 06695100 0
1477 4809 0C41 003C 00F0 06696100 0
1478 4809 0C41 003C 00F0 06697100 0
1479 5000 0070 0070 0060 06698100 0
1480 4809 0C41 003C 00F0 06699100 0
1481 4809 0C41 003C 00F0 06700100 0
1482 5000 0070 0070 0060 06701100 0
1483 4809 0C41 003C 00F0 06702100 0
1484 4809 0C41 003C 00F0 06703100 0
1485 5000 0070 0070 0060 06704100 0
1486 4809 0C41 003C 00F0 06705100 0
1487 4809 0C41 003C 00F0 06706100 0
1488 5000 0070 0070 0060 06707100 0
1489 4809 0C41 003C 00F0 06708100 0
1490 4809 0C41 003C 00F0 06709100 0
1491 5000 0070 0070 0060 06710100 0
1492 4809 0C41 003C 00F0 06711100 0
1493 4809 0C41 003C 00F0 06712100 0
1494 4809 0C41 003C 00F0 06713100 0
1495 5000 0070 0070 0060 06714100 0
1496 4809 0C41 003C 00F0 06715100 0
1497 4809 0C41 003C 00F0 06716100 0
1498 5000 0070 0070 0060 06717100 0

```

```

OPCODE46:
1460 5F9C 0000 0000 0000 0
1461 3000 0000 0000 0000 0
1462 1E10 0000 0000 0000 0
1463 4E10 0000 0000 0000 0
1464 1E10 0000 0000 0000 0
1465 4E10 0000 0000 0000 0
1466 1E10 0000 0000 0000 0
1467 4E10 0000 0000 0000 0
1468 1E10 0000 0000 0000 0
1469 4E10 0000 0000 0000 0
1470 1E10 0000 0000 0000 0
1471 2260 0000 0000 0000 0
1472 2260 0000 0000 0000 0
1473 2260 0000 0000 0000 0
1474 2260 0000 0000 0000 0
1475 2260 0000 0000 0000 0
1476 2260 0000 0000 0000 0
1477 2260 0000 0000 0000 0
1478 2260 0000 0000 0000 0
1479 2260 0000 0000 0000 0
1480 2260 0000 0000 0000 0
1481 2260 0000 0000 0000 0
1482 2260 0000 0000 0000 0
1483 2260 0000 0000 0000 0
1484 2260 0000 0000 0000 0
1485 2260 0000 0000 0000 0
1486 2260 0000 0000 0000 0
1487 2260 0000 0000 0000 0
1488 2260 0000 0000 0000 0
1489 2260 0000 0000 0000 0
1490 2260 0000 0000 0000 0
1491 2260 0000 0000 0000 0
1492 2260 0000 0000 0000 0
1493 2260 0000 0000 0000 0
1494 2260 0000 0000 0000 0
1495 2260 0000 0000 0000 0
1496 2260 0000 0000 0000 0
1497 2260 0000 0000 0000 0
1498 2260 0000 0000 0000 0
1499 2260 0000 0000 0000 0

```


OPCODE	OPCODE47:	OPCODE47F:	OPCODE4701:
0000 0000 0000 0020	COMP 16 = SAR		
B			
0001 0000 0000 0000	IF HST THEN STEP ELSE SKIP		
4809 0000 0000 0000	BUMP - 1 = HPCR		
480B 0000 0000 0000	15 = LIT		
480C 0000 0000 0000	A3 AND LIT = A3		
480D 0000 0000 0000	A3 EOL		
480E 0000 0000 0000	IF TRUE THEN 0 = B1 SKIP		
480F 0000 0000 0000	CONTENTSMR - 1 = CPCR		
4810 0000 0000 0000	0 = A3		
4811 0000 0000 0000	IFETCH - 1 = CPCR		
4812 0000 0000 0000	A3 + B = B		
4813 0000 0000 0000	A1 R = A1		
4814 0000 0000 0000	COMP 16 = SAR		
4815 0000 0000 0000	A1 L = A1		
4816 0000 0000 0000	A1 OR B = A1		
4817 0000 0000 0000	OPCODE - 1 = HPCR		
0001 0000 0000 0000			
0002 0000 0000 0000			
0003 0000 0000 0000			
0004 0000 0000 0000			
0005 0000 0000 0000			
0006 0000 0000 0000			
0007 0000 0000 0000			
0008 0000 0000 0000			
0009 0000 0000 0000			
0010 0000 0000 0000			
0011 0000 0000 0000			
0012 0000 0000 0000			
0013 0000 0000 0000			
0014 0000 0000 0000			
0015 0000 0000 0000			
0016 0000 0000 0000			
0017 0000 0000 0000			
0018 0000 0000 0000			
0019 0000 0000 0000			
0020 0000 0000 0000			
0021 0000 0000 0000			
0022 0000 0000 0000			
0023 0000 0000 0000			
0024 0000 0000 0000			
0025 0000 0000 0000			
0026 0000 0000 0000			
0027 0000 0000 0000			
0028 0000 0000 0000			
0029 0000 0000 0000			
0030 0000 0000 0000			
0031 0000 0000 0000			
0032 0000 0000 0000			
0033 0000 0000 0000			
0034 0000 0000 0000			
0035 0000 0000 0000			
0036 0000 0000 0000			
0037 0000 0000 0000			
0038 0000 0000 0000			
0039 0000 0000 0000			
0040 0000 0000 0000			
0041 0000 0000 0000			
0042 0000 0000 0000			
0043 0000 0000 0000			
0044 0000 0000 0000			
0045 0000 0000 0000			
0046 0000 0000 0000			
0047 0000 0000 0000			
0048 0000 0000 0000			
0049 0000 0000 0000			
0050 0000 0000 0000			
0051 0000 0000 0000			
0052 0000 0000 0000			
0053 0000 0000 0000			
0054 0000 0000 0000			
0055 0000 0000 0000			
0056 0000 0000 0000			
0057 0000 0000 0000			
0058 0000 0000 0000			
0059 0000 0000 0000			
0060 0000 0000 0000			
0061 0000 0000 0000			
0062 0000 0000 0000			
0063 0000 0000 0000			
0064 0000 0000 0000			
0065 0000 0000 0000			
0066 0000 0000 0000			
0067 0000 0000 0000			
0068 0000 0000 0000			
0069 0000 0000 0000			
0070 0000 0000 0000			
0071 0000 0000 0000			
0072 0000 0000 0000			
0073 0000 0000 0000			
0074 0000 0000 0000			
0075 0000 0000 0000			
0076 0000 0000 0000			
0077 0000 0000 0000			

1484	4809	CC41	0C70	C0F0	B L = B	% SHIFT (RCA) INTO UHW OF B	06778C00 D
1485	0000	00C3	0C70	F02C	COMP 16 = SAR	% PUT B IN THE ADDR, TEST (RCA) < 0	06779C00 D
1486	4809	0C40	0000	00F0	IF RST THEN SKIP		06706C00 D
1487	4819	CC03	007C	C0FC	OPCODE - 1 = MPCR		067F8C00 D
1488	56EC	0C03	0C7C	C0FC	CONTENTSRM - 1 = CPCR	% (RCH) INTO D	067F2C00 D
1489	2380	0000	0000	006C	CONTENTSRM - 1 = CPCR	% CLEAR LOWER 16 BITS OF A1	06783C00 D
1490	4809	AC03	CC50	00FC	COMP 16 = SAR		06794C00 D
1491	0000	00C0	0C7C	F020	A1 L = A1		06785C00 D
1492	4809	AC01	4C20	00F0	A1 OR B = A1	% MODIFY PAR 10 JUMP	06786C00 D
1493	4809	AC5E	4070	C0FC	OPCODE - 1 = MPCR		06787C00 C
1494	56EC	0000	0030	C040			06788C00 D
1495	2809	0000	0C3C	C0F0	IF LC1	% RI TYPE 1 LOCAL JUMP LESS INSTRUCTION	06791C00 C
1496	226C	C0C0	0020	C06C	CHECKCC - 1 = CPCR	% IF (CC) = 11, (P) + 0 = 1	06792C00 D
1497	4809	2E52	000C	C0F0	LIT F0L B	% PUT THE CC BITS INTO B	06793C00 D
1498	003C	C0C3	000C	00E0	3 = LIT	% CHECK CC FOR 11	06794C00 D
1499	66CB	C0C0	C030	00F0	IF FALSE THEN STEP ELSE SKIP		06795C00 D
1500	66EC	00C0	C070	0040	OPCODE - 1 = MPCR		06796C00 D
1501	5590	0000	0020	C04C	R11 - 1 = MPCR	% (P) + 0 = P	06797C00 D
1502	228C	00C3	000C	0060	CONTENTSRM - 1 = CPCR	% RK TYPE JUMP NEGATIVE	06800C00 D
1503	4809	0C40	0000	0020	B L = B	% IF (RCA) < 0, (Y) = P	06801C00 D
1504	0000	00C0	0000	0020	COMP 16 = SAR	% (RCH) INTO B	06802C00 D
1505	4809	0C40	0000	0020	IF RST THEN SKIP	% SHIFT (RCA) INTO UHW OF B	06803C00 D
1506	4819	CC03	007C	C0FC	CONTENTSRM - 1 = CPCR	% CHECK IF (RCA) < 0	06804C00 D
1507	2380	0000	0000	006C	CONTENTSRM - 1 = MPCR	% ADVANCE THE PAR BY 1 AND CALL OPCODE	06805C00 D
1508	4809	0C40	0000	0020	B L = B	% (RCH) INTO E	06806C00 D
1509	4809	0C40	0000	0020	IF RST THEN SKIP	% PUT Y INTO B	06807C00 C
1510	4809	0C40	0000	0020	IF RST THEN SKIP	% PUT PAR VALUE IN LS WORD OF B	06808C00 D
1511	4809	0C40	0000	0020	IF RST THEN SKIP	% CLEAR OLD PAR	06809C00 D
1512	4809	0C40	0000	0020	IF RST THEN SKIP	% CONSTRUCT NEW PAR	06810C00 D
1513	4809	0C40	0000	0020	IF RST THEN SKIP	% RX TYPE JUMP INSTRUCTION	06811C00 D
1514	4809	0C40	0000	0020	IF RST THEN SKIP	% IF (RCA) < 0, (Y) = P	06812C00 D
1515	4809	0C40	0000	0020	IF RST THEN SKIP	% SHIFT "A" INTO LS BITS OF B	06813C00 D
1516	4809	0C40	0000	0020	IF RST THEN SKIP	% ISOLATE "A" INTO B	06814C00 D
1517	4809	0C40	0000	0020	IF RST THEN SKIP	% (RCA) INTO MAR2	06815C00 D
1518	4809	0C40	0000	0020	IF RST THEN SKIP	% (RCA) INTO E	06816C00 D
1519	4809	0C40	0000	0020	IF RST THEN SKIP	% (RCA) SHIFTED INTO UHW OF B	06817C00 D
1520	4809	0C40	0000	0020	IF RST THEN SKIP	% CHECK IF (RCA) < 0	06818C00 C
1521	4809	0C40	0000	0020	IF RST THEN SKIP	% ADVANCE THE PAR BY 1 AND CALL OPCODE	06819C00 D
1522	4809	0C40	0000	0020	IF RST THEN SKIP	% RESTORE INSTRUCTION IN E	06820C00 D
1523	4809	0C40	0000	0020	IF RST THEN SKIP	% ANALYZE "M" FIELD	06821C00 D
1524	4809	0C40	0000	0020	IF RST THEN SKIP	% PUT Y INTO PAR2	06822C00 D
1525	4809	0C40	0000	0020	IF RST THEN SKIP	% PUT (Y) INTO B	06823C00 D
1526	4809	0C40	0000	0020	IF RST THEN SKIP		06824C00 D
1527	4809	0C40	0000	0020	IF RST THEN SKIP		06825C00 D
1528	4809	0C40	0000	0020	IF RST THEN SKIP		06826C00 D
1529	4809	0C40	0000	0020	IF RST THEN SKIP		06827C00 D
1530	4809	0C40	0000	0020	IF RST THEN SKIP		06828C00 D
1531	4809	0C40	0000	0020	IF RST THEN SKIP		06829C00 D
1532	4809	0C40	0000	0020	IF RST THEN SKIP		06830C00 D
1533	4809	0C40	0000	0020	IF RST THEN SKIP		06831C00 D
1534	4809	0C40	0000	0020	IF RST THEN SKIP		06832C00 D
1535	4809	0C40	0000	0020	IF RST THEN SKIP		06833C00 D
1536	4809	0C40	0000	0020	IF RST THEN SKIP		06834C00 D
1537	4809	0C40	0000	0020	IF RST THEN SKIP		06835C00 D
1538	4809	0C40	0000	0020	IF RST THEN SKIP		06836C00 D
1539	4809	0C40	0000	0020	IF RST THEN SKIP		06837C00 D
1540	4809	0C40	0000	0020	IF RST THEN SKIP		06838C00 D
1541	4809	0C40	0000	0020	IF RST THEN SKIP		06839C00 D
1542	4809	0C40	0000	0020	IF RST THEN SKIP		06840C00 D
1543	4809	0C40	0000	0020	IF RST THEN SKIP		06841C00 D
1544	4809	0C40	0000	0020	IF RST THEN SKIP		06842C00 D
1545	4809	0C40	0000	0020	IF RST THEN SKIP		06843C00 D
1546	4809	0C40	0000	0020	IF RST THEN SKIP		06844C00 D
1547	4809	0C40	0000	0020	IF RST THEN SKIP		06845C00 D
1548	4809	0C40	0000	0020	IF RST THEN SKIP		06846C00 D
1549	4809	0C40	0000	0020	IF RST THEN SKIP		06847C00 D
1550	4809	0C40	0000	0020	IF RST THEN SKIP		06848C00 D
1551	4809	0C40	0000	0020	IF RST THEN SKIP		06849C00 D
1552	4809	0C40	0000	0020	IF RST THEN SKIP		06850C00 D
1553	4809	0C40	0000	0020	IF RST THEN SKIP		06851C00 D
1554	4809	0C40	0000	0020	IF RST THEN SKIP		06852C00 D
1555	4809	0C40	0000	0020	IF RST THEN SKIP		06853C00 D
1556	4809	0C40	0000	0020	IF RST THEN SKIP		06854C00 D
1557	4809	0C40	0000	0020	IF RST THEN SKIP		06855C00 D
1558	4809	0C40	0000	0020	IF RST THEN SKIP		06856C00 D
1559	4809	0C40	0000	0020	IF RST THEN SKIP		06857C00 D
1560	4809	0C40	0000	0020	IF RST THEN SKIP		06858C00 D
1561	4809	0C40	0000	0020	IF RST THEN SKIP		06859C00 D
1562	4809	0C40	0000	0020	IF RST THEN SKIP		06860C00 D
1563	4809	0C40	0000	0020	IF RST THEN SKIP		06861C00 D
1564	4809	0C40	0000	0020	IF RST THEN SKIP		06862C00 D
1565	4809	0C40	0000	0020	IF RST THEN SKIP		06863C00 D
1566	4809	0C40	0000	0020	IF RST THEN SKIP		06864C00 D
1567	4809	0C40	0000	0020	IF RST THEN SKIP		06865C00 D
1568	4809	0C40	0000	0020	IF RST THEN SKIP		06866C00 D
1569	4809	0C40	0000	0020	IF RST THEN SKIP		06867C00 D
1570	4809	0C40	0000	0020	IF RST THEN SKIP		06868C00 D
1571	4809	0C40	0000	0020	IF RST THEN SKIP		06869C00 D
1572	4809	0C40	0000	0020	IF RST THEN SKIP		06870C00 D
1573	4809	0C40	0000	0020	IF RST THEN SKIP		06871C00 D
1574	4809	0C40	0000	0020	IF RST THEN SKIP		06872C00 D
1575	4809	0C40	0000	0020	IF RST THEN SKIP		06873C00 D
1576	4809	0C40	0000	0020	IF RST THEN SKIP		06874C00 D
1577	4809	0C40	0000	0020	IF RST THEN SKIP		06875C00 D
1578	4809	0C40	0000	0020	IF RST THEN SKIP		06876C00 D
1579	4809	0C40	0000	0020	IF RST THEN SKIP		06877C00 D
1580	4809	0C40	0000	0020	IF RST THEN SKIP		06878C00 D
1581	4809	0C40	0000	0020	IF RST THEN SKIP		06879C00 D
1582	4809	0C40	0000	0020	IF RST THEN SKIP		06880C00 D
1583	4809	0C40	0000	0020	IF RST THEN SKIP		06881C00 D
1584	4809	0C40	0000	0020	IF RST THEN SKIP		06882C00 D
1585	4809	0C40	0000	0020	IF RST THEN SKIP		06883C00 D
1586	4809	0C40	0000	0020	IF RST THEN SKIP		06884C00 D
1587	4809	0C40	0000	0020	IF RST THEN SKIP		06885C00 D
1588	4809	0C40	0000	0020	IF RST THEN SKIP		06886C00 D
1589	4809	0C40	0000	0020	IF RST THEN SKIP		06887C00 D
1590	4809	0C40	0000	0020	IF RST THEN SKIP		06888C00 D
1591	4809	0C40	0000	0020	IF RST THEN SKIP		06889C00 D
1592	4809	0C40	0000	0020	IF RST THEN SKIP		06890C00 D
1593	4809	0C40	0000	0020	IF RST THEN SKIP		06891C00 D
1594	4809	0C40	0000	0020	IF RST THEN SKIP		06892C00 D
1595	4809	0C40	0000	0020	IF RST THEN SKIP		06893C00 D
1596	4809	0C40	0000	0020	IF RST THEN SKIP		06894C00 D
1597	4809	0C40	0000	0020	IF RST THEN SKIP		06895C00 D
1598	4809	0C40	0000	0020	IF RST THEN SKIP		06896C00 D
1599	4809	0C40	0000	0020	IF RST THEN SKIP		06897C00 D
1600	4809	0C40	0000	0020	IF RST THEN SKIP		06898C00 D
1601	4809	0C40	0000	0020	IF RST THEN SKIP		06899C00 D
1602	4809	0C40	0000	0020	IF RST THEN SKIP		06900C00 D
1603	4809	0C40	0000	0020	IF RST THEN SKIP		06901C00 D
1604	4809	0C40	0000	0020	IF RST THEN SKIP		06902C00 D
1605	4809	0C40	0000	0020	IF RST THEN SKIP		06903C00 D
1606	4809	0C40	0000	0020	IF RST THEN SKIP		06904C00 D
1607	4809	0C40	0000	0020	IF RST THEN SKIP		06905C00 D
1608	4809	0C40	0000	0020	IF RST THEN SKIP		06906C00 D
1609	4809	0C40	0000	0020	IF RST THEN SKIP		06907C00 D
1610	4809	0C40	0000	0020	IF RST THEN SKIP		06908C00 D
1611	4809	0C40	0000	0020	IF RST THEN SKIP		06909C00 D
1612	4809	0C40	0000	0020	IF RST THEN SKIP		06910C00 D
1613	4809	0C40	0000	0020	IF RST THEN SKIP		06911C00 D
1614	4809	0C40	0000	0020	IF RST THEN SKIP		06912C00 D
1615	4809	0C40	0000	0020	IF RST THEN SKIP		06913C00 D
1616	4809	0C40	0000	0020	IF RST THEN SKIP		06914C00 D
1617	4809	0C40	0000	0020	IF RST THEN SKIP		06915C00 D
1618	4809	0C40	0000	0020	IF RST THEN SKIP		06916C00 D
1619	4809	0C40	0000	0020	IF RST THEN SKIP		06917C00 D
1620	4809	0C40	0000	0020	IF RST THEN SKIP		

14E4	8809 A003 C030 00F0	A1 R = A1	% CLEAR THE OLD PAR	06836100 0
14E5	0000 0000 0000 0020	COMP 16 = SAR		06839100 0
14E6	8809 A001 40FF 00F0	A1 L = A1	% PUT (Y) INTO THE PAR	06840100 0
14E7	8809 AC5C 4000 00F0	A1 OR B = A1		06841100 0
14E8	66EC 0000 0000 0E4C	OPCODE - 1 = MPCR		06842100 0
			%	06843100 0
			% NOT IMPLEMENTED	06844100 0
14E9	4E5C 0000 0000 0040	NOTIMP - 1 = MPCR		06845000 0
			%	06846100 0
14EA	4E5C 0000 0000 0040	OPCODE51: NOTIMP - 1 = MPCR		06847100 0
			% NOT IMPLEMENTED	06848100 0
14EB	4E5D 0003 0000 004C	OPCODE52: NOTIMP - 1 = MPCR		06849100 0
			% NOT IMPLEMENTED	06850100 0
14EC	4E5D 0000 0000 0040	OPCODE53: NOTIMP - 1 = MPCR		06851000 0
			%	06852100 0
			% NOTIMPLEMENTED	06853100 0
			%	06854100 0
			% LOAD ADDRESS REGISTER(S)	06855100 0
14ED	5F9C 00C3 0000 C06C	YFCODE - 1 = CPCR		06856100 0
14EE	8809 C640 0000 00F0	A2 + AMPCR = AMPCR		06857100 0
14EF	1EBC 0003 C030 00C0	OP54F - 1 = AMPCR		06858100 0
14F0	8809 0000 0000 00F0	STEP		06859100 0
14F1	4824 0000 0000 00F0	EXEC		06860100 0
14F2	1ECC 00C0 0000 0040	OP540 - 1 = MPCR		06861100 0
14F3	1E00 C003 0000 0040	OP541 - 1 = MPCR		06862100 0
14F4	3000 00C0 0000 0040	FAULT - 1 = MPCR		06863100 0
14F5	1E00 0000 0000 0040	OP543 - 1 = MPCR		06864100 0
			%	06865100 0
14F6	4809 E155 0000 00F0	A3 AND LIT = B		06866100 0
14F7	00FC 00C0 0000 C0C0	15 = LIT		06867100 0
14F8	0000 0000 0000 006C	REGSTACK - 1 = CPCR		06868100 0
14F9	2100 0000 0000 0060	CINPUL - 1 = CPCR		06869100 0
14FA	4809 0000 0000 C0F0	B = M1R		06870100 0
			%	06871100 0
			% LAR: LOAD ADDRESS REGS: TYPE RR	06872100 0
14FB	8809 E000 8000 00F0	A3 R = D		06873100 0
14FC	30F0 0000 0000 0000	4 = SARI, 15 = LIT		06874100 0
14FD	8809 2C56 C800 C0F0	LIT AND R = B		06875100 0
14FE	51C0 0003 0000 0060	REGSTACK - 1 = CPCR		06876100 0
14FF	2F3C 0000 0000 C66C	CINPUL - 1 = CPCR		06877100 0
1500	48C9 2C56 2000 00F0	LIT AND B = A2		06878100 0
1501	00FC 00C0 0000 00E0	255 = LIT		06879100 0
1502	4809 C643 001C 00F0	A2 + AMPCR = MAR2		06880100 0
1503	3000 0000 0000 00C0	PAGESES = AMPCR		06881100 0
1504	8809 0000 0000 00C0	STEP		06882100 0
1505	2F50 00C0 0000 C06C	OPCODE - 1 = CPCR		06883100 0
1506	66EC 00C3 0000 0040	EOUTPUT - 1 = MPCR		06884100 0
			%	06885100 0
			% SEPARATE AMPCR ASSIGNMENTS	06886100 0
			%	06887100 0
			%	06888100 0
			% LAR: LOAD ADDRESS REGISTER (INDIRECT)	06889100 0
1507	4809 E156 0000 00F0	(Y*) -> AR R		06890100 0
1508	00FC 00C0 0000 C06C	15 = LIT		06891100 0
1509	51C0 0003 0000 0060	REGSTACK - 1 = CPCR		06892100 0
150A	2100 0000 0000 0060	CINPUL - 1 = CPCR		06893100 0
150B	4809 0000 0000 00C0	0 LIT = BR2		06894100 0
150C	0000 C0C3 0000 C030	COMP 8 = SAR		06895100 0
			% B = (R(M)) = Y*	06896100 0
			% Y* INTO BR2	06897100 0


```

1598 4809 0C41 1000 00F0
1599 001C 0000 00F0 00F0
159A 4809 2F5C 801C 00F0
159B 2F30 00C0 00C0 00F0
159C 4809 EC5C 100C 00F0
159D 4809 CFC0 00C0 00F0
159E 49C9 E000 987C 00F0
159F 2000 0000 00C0 00F0
15A0 4809 E001 0820 00F0
15A1 0000 00C0 00C0 00F0
15A2 4809 0C40 303F 00F0
15A3 2F50 00C0 0000 00F0
15A4 4809 00C0 00F0 20F0
15A5 4809 0F43 801C 00F0
15A6 2F3C 00C0 00C0 00F0
15A7 4809 00F0 00F0 00F0
15A8 0000 00C0 00C0 00F0
15A9 2F5C 00C0 0000 00F0
15AA 56E0 0003 0F2C 00F0

15AB 4809 2C55 7020 00F0
15AC 00F0 C500 0C20 00F0
15AD 28C9 C040 885F 00F0
15AE 80FC 00C0 C07F 00F0
15AF 4809 2C56 76D0 00F0
15B0 51C0 0F00 0020 00F0
15B1 4809 0F41 0020 00F0
15B2 0000 00C0 0C30 00F0
15B3 2F3C 00C0 00C0 00F0
15B4 4809 0C41 0E30 00F0
15B5 0000 0000 003C 00F0
15B6 4809 0C40 0000 00F0
15B7 48C9 C000 00C0 00F0
15B8 0C10 0000 00C0 00F0
15B9 4809 C5C0 100C 00F0
15BA 2F3C 00C0 00C0 00F0
15BB 2F3C 00C0 00C0 00F0
15BC 4809 E0C0 90C0 00F0
15BD 0000 0F20 00C0 00F0
15BE 4809 E001 1070 00F0
15BF 4809 E05C 1030 00F0
15C0 4809 C0C0 00C0 80F0
15C1 4809 E0D0 9030 00F0
15C2 4809 00C0 0800 00F0
15C3 2C09 0C19 0920 00F0
15C4 4809 0C41 C800 00F0
15C5 49C9 EC5C 1E30 00F0
15C6 2000 00C0 00C0 00F0
15C7 4809 E001 2F20 00F0
15C8 0000 00C0 007C 00F0
15C9 4809 CFC0 8030 00F0
15CA 2F50 00C0 00C0 00F0
15CB 4809 00C0 0C3C 2C0F
15CC 4809 CFC0 801C 00F0
15CD 0000 0000 0000 00F0
15CE 48C9 E0F0 805F 00F0

B L = A3
16 = SAR J 1 = LIT
LIT OR BHAR = HAR2
EINPUI - 1 = CPCK
A3 OR B = A3
A2 = SAR
A3 R = A3,BJ SET LC1
SETCCA - 1 = CPCK
A3 L = B
COMP 16 = SAR
B R = HAR
EINPUI - 1 = CPCK
ASR OR BHAR = HAR2
BHAR R = HAR2
B = SAR
A3 = HAR
EINPUI - 1 = CPCK
OPCODE - 1 = MPCK

% RL TYPE ALG RIGHT DOUBLE SHIFT
% RIGHT SHIFT (R(A+1)) M (C-3)
% SIGN FILL AND SET CC
% ISOLATE *H* FIELD

% PUT B IN THE ADDER
% NEG SIGN FLAG
% A3 = (R(A))/INSTRUCTION
% R(A+1)
% R(A+1) INTO B
% CLEAR LHM OF A3
% A3 = (R(A))/(R(A+1))
% PERFORM SHIFT

% B = 1111/0000 OR B = 0000/0000
% SET THE (CONDITION BITS)
% (R(A+1)) SHIFTED INTO MIR
% REFERENCE 961
% R(A)

% RL TYPE ALG RIGHT DOUBLE SHIFT
% RIGHT SHIFT (R(A+1)) M (C-3)
% SIGN FILL AND SET CC
% ISOLATE *H* FIELD

% PUT B IN THE ADDER
% NEG SIGN FLAG
% A3 = (R(A))/INSTRUCTION
% R(A+1)
% R(A+1) INTO B
% CLEAR LHM OF A3
% A3 = (R(A))/(R(A+1))
% PERFORM SHIFT

% B = 1111/0000 OR B = 0000/0000
% SET THE (CONDITION BITS)
% (R(A+1)) SHIFTED INTO MIR
% REFERENCE 961
% R(A)

```

0F603:


```

15CF 30C0 00C0 0000 00C0      16 = SAR
15D0 2F5C 0000 0000 00A0      EUIPUT - 1 = CPCR
15E1 66EC 00C0 0000 00A0      OPCODE - 1 = MPCR

OPCODE61: XFCD0E - 1 = CPCR
A2 + AMPCR = AMPCR
0P61F - 1 = AMPCR
STEP
EXEC

15D6 5F9C 00C0 0000 006C      0P610:
15E7 4809 C640 00C0 006C      0P611:
15F8 4809 C640 00C0 006C      15 = LIT 4 = SAR
15F9 4809 C640 00C0 006C      P R = B
15FA 4809 C640 00C0 006C      REGSTACK - 1 = CPCR
15FB 4809 C640 00C0 006C      EUIPUT - 1 = CPCR
15FC 4809 C640 00C0 006C      B = A3
15FD 4809 C640 00C0 006C      A3 L = A3
15FE 4809 C640 00C0 006C      COMP 16 = SAR

15D0 4809 2C56 20C0 006C      15 = LIT 4 = SAR
15D1 80C0 C6C0 00C0 006C      P R = B
15D2 4809 0C40 80C0 006C      REGSTACK - 1 = CPCR
15D3 4809 2C56 00C0 006C      EUIPUT - 1 = CPCR
15D4 2F3C 00C0 00C0 006C      B = A3
15D5 4809 0C40 1C70 006C      A3 L = A3
15D6 4809 E0C1 10C0 006C      COMP 16 = SAR
15D7 0100 0000 0000 006C      A3 OR B = A3
15D8 4809 EC5C 10C0 006C      A2 = B
15D9 4809 C000 70C0 006C      LIT - B = A2
15DA 4809 2C56 20C0 006C      A2 = SAR
15DB 4809 C000 0000 006C      A3 C = A3
15DC 4809 E0C1 90C0 006C      COMP 16 = SAR
15DD 4809 E0C1 4C00 006C
15DE 00C0 00C0 00C0 006C

```



```

1632 1809 C0C0 0C30 80F0
1633 19C9 E001 9830 C0F0
1634 20C0 0000 0C9C 006C
1635 1809 E001 0830 00F0
1636 0000 0000 0000 C020
1637 1809 0C40 8C30 C0F0
1638 2F50 0000 0C9C 006C
1639 4809 0000 0C90 20F0
163A 4809 6F43 801C 00F0
163B 0000 C0C0 0C30 0010
163C 1809 E000 803C 00FC
163D 0000 0000 0C9C 0020
163E 2F5C 0000 0C90 C060
163F 56E0 0000 0C30 0040

1640 5C9C C0C0 0030 0060
1641 4809 C643 0C30 00F0
1642 1F0C 0000 C030 00F0
1643 4809 0000 0C3C 00FC
1644 4824 C000 0C30 0010

1645 1FEC 0000 007C C090
1646 1FEC 0000 0C30 004C
1647 2000 0000 0000 004C
1648 2010 0000 0C00 004C

1649 228C C0C0 0C90 C060
164A 4809 C041 2C3C 00FC
164B 00F0 00C0 0C3C 00A0
164C 4830 E156 0830 C0F0
164D 4809 0C41 0C30 C0F0
164E 4849 CCCE 0C30 00FC
164F 78C9 0000 C03C 00FC
1650 1E70 C0C0 0C30 006C
1651 4F1C 0C3C CC90 C060
1652 4809 00C0 003C 00FC
1653 4809 0C40 5B3C 00F0
1654 2000 0000 0C3C 002C
1655 200C 0000 0C30 0060
1656 2F5C C0C0 003C 004C
1657 56EC 0000 007C C09C

1658 4809 2C5C 0C30 00F0
1659 9CFC 00C0 0C30 0080
165A 4809 0C40 8B3C 00FC
165B 4809 2C5C 083C 00F0
165C 51EC 0000 0C3C 0060
165D 4809 0F41 0C20 C0F0
165E 000C 0000 0C30 0030
165F 2F30 C0C0 0C30 0060
1660 4809 0C41 2C00 C0F0
1661 001C 0C3C 0C90 0040

A2 + 1 = SAR
A3 C = A340J SET LC1
SETCCA - 1 = CPCR
A3 L = B
COMP 16 = SAR
B R = MIR
[OUTPUT] - 1 = CPCR
ASR
RHAR R = HAR2
B = SAR
A3 R = MIR
16 = SAR
E0UTPUT - 1 = CPCR
OPCODE - 1 = MPCR

OPCODE62: XFCODE - 1 = CPCR
A2 + AMPCR = AMPCR
OP62F - 1 = AMPCR
STEP
EXEC

OP62F1
OP620 - 1 = MPCR
OP621 - 1 = MPCR
OP622 - 1 = MPCR
OP623 - 1 = MPCR

OP620:
CONTENTSNA - 1 = CPCR
E L = A2
COMP 16 = SAP; 15 = LIT
A3 AND LIT = B
B L = B
A2 - B = MIR; SET LC2
IF A0V THEN SET LC1
CARRY - 1 = CPCR
CHECKOV - 1 = CPCR
RMI
B R = MIR#B
16 = SAR
SETCCA - 1 = CPCR
E0UTPUT - 1 = CPCR
OPCODE - 1 = MPCR

OP621:
LIT AND B = MIR
15 = LIT; B = SAR
B R = R
LIT AND B = 0
REGSTACK - 1 = CPCR
RHAR L = BR1
COMP B = SAR
FINPUT - 1 = CPCR
B L = A2
CMP 16 = SAR; 1 = LIT

```



```

14C4 2809 0C40 8030 C0FC
14C5 9CFC 0C00 CF70 00B0
14C6 9809 2C56 0000 C0F0
14C7 51C0 0C00 0000 C060
14C8 4809 CF41 CF70 00FC
14C9 0000 0000 0000 0030
14CA 4809 0F45 0B30 C0FC
14CB 51C0 0C00 0030 C060
14CC 2F30 00C0 CC70 C060
14CD 4809 E155 2000 00F0
14CE 40FC 0000 0000 00E0
14CF 4C60 0000 0030 C060
14D0 4809 0F00 0B30 00F0
14D1 2000 C000 0000 C060
14D2 4809 EC01 0B30 00F0
14D3 00C0 0000 CC70 C020
14D4 4809 CC40 8630 C0FC
14D5 2F50 C0C0 CC70 C060
14D6 4809 0C00 0000 00F0
14D7 4809 0F43 861C 00F0
14D8 0000 00C0 0000 0010
14D9 4809 0F00 8630 00F0
14DA 0000 0000 0030 0020
14DB 2F50 C0C0 0000 C06C
14DC 5650 0000 0030 C040

```

```

% TYPE RL LITERAL MULTIPLY
% R(A+1) X M = (R(A)*R(A+1)), SET CC

B R = B; IF LCI
4 = SARI 15 = LII
LII AND B = B
REGSTACK - 1 = CPGR
PHAR L = BR1
COMP 8 = SAR
BHAR + 1 = B
REGSTACK - 1 = CHCR
EINPUT - 1 = CPGR
A3 AND LII = A2
15 = LIT
MULT - 1 = CPGR
A3 = B; MIR
SEITCCA - 1 = CPGR
A3 L = B
COMP 16 = SAR
EINPUT - 1 = CPGR
ASR
BHAR R = HAR2
8 = SAR
A3 R = MIR
16 = SAR
EINPUT - 1 = CPGR
OPCODE - 1 = HPGR

% TYPE RL LITERAL DIVIDE
% (R(A)*R(A+1))/H = R(A+1), REMAINDER
% INTO R(A) SET CC AND OV
% ISOLATE "H"

LII AND 0 = B
15 = LIT; COMP 16 = SAR
B L = MIR
A3 R = R
4 = SAR
LII AND 0 = E
REGSTACK - 1 = CPGR
BHAR L = UR1
COMP 8 = SAR
EINPUT - 1 = CPGR
R L = A2
COMP 16 = SAR
BHAR + 1 = B
REGSTACK - 1 = CPGR
EINPUT - 1 = CPGR
A2 OR B = A2; BHI; SET LCI
A3 - 1 = CPGR
PHAR R = HAR2
8 = SAR
A2 = MIR
EINPUT - 1 = CPGR
A3 = B; MIR
BHAR + 1 = B
REGSTACK - 1 = CPGR
EINPUT - 1 = CPGR
CLEAROV - 1 = CPGR

% TYPE RL LITERAL MULTIPLY
% R(A+1) X M = (R(A)*R(A+1)), SET CC

% ISOLATE "A"

% TEMP STORAGE

% R(A+1)
% R(A+1) INTO HAR2
% (R(A+1)) INTO B
% "H" INTO A2

% (R(A+1)) X H INTO A3
% PRODUCT INTO B; MIR
% SET THE CONDITION BITS

% (R(A+1))

% REF FRI
% R(A)

% (R(A))

% TYPE RL LITERAL DIVIDE
% (R(A)*R(A+1))/H = R(A+1), REMAINDER
% INTO R(A) SET CC AND OV
% ISOLATE "H"

% LEFT JUSTIFY DIVISOR

% ISOLATE "A"
% R(A) INTO HAR2
% STORE R(A)

% (R(A)) INTO E

% PCA+1)
% F(A+1) INTO HAR2
% (R(A+1)) INTO B
LCI X A2 = (H(A))/R(A+1)
% (R(A)*R(A+1))/H
% REF FRI
% R(A)

% REMAINDER

% QUOTIENT INTO B; MIR
% SET THE CONDITION CODE
% R(A+1)
% R(A+1) INTO HAR2
% QUOTIENT INTO R(A+1)
% SET OV BIT

```

```

14DD 4809 2C55 0B00 C0AC
14DE 00FC C0C3 0C7C 00AC
14DF 48C9 0C41 0C50 00F0
14E0 4809 0C00 8630 00FC
14E1 500C C0C0 0030 00C0
14E2 4809 2C56 CB40 C0F0
14E3 51C0 C0C0 0000 C06C
14E4 4809 0F41 0C20 00F0
14E5 0000 00C0 0000 0030
14E6 2F3C 00C0 0000 0060
14E7 48C9 CC41 207C 00F0
14E8 00C0 00C0 0000 C02C
14E9 4809 0F46 C030 C0FC
14EA 51C0 C0C0 0000 C060
14EB 2F50 C000 0000 C06C
14EC 0000 0000 0000 0000
14ED 2400 0000 0000 0000
14EE 4809 0000 0030 C0F0
14EF 4809 0F43 861C 00F0
14F0 0000 C0C3 0000 0010
14F1 4809 C000 0F40 00FC
14F2 2F50 C0C0 0000 0060
14F3 4809 EC00 0B30 00FC
14F4 2000 0000 0000 C060
14F5 4809 0F45 0B30 00F0
14F6 51C0 C0C0 0030 C060
14F7 2F50 C0C0 C030 C060
14F8 5050 0000 C000 C06C

```



```

172A 2080 00C0 0000 0040
172B 4849 0003 0030 00F0
172C 570C 0000 00C0 0060
172D 4809 0C41 0010 00F0
172E 500C 00C0 0030 0030
172F 500C 00C0 0030 0060
1730 4809 0C41 0030 00F0
1731 000C 00C0 00C0 0020
1732 4809 0C00 0030 00F0
1733 4809 0C41 0030 0060
1734 4809 0C41 0030 00F0
1735 00C0 0000 0030 0020
1736 3808 0C00 00C0 00F0
1737 20C0 00C0 0000 0040
1738 4809 0C40 80C0 00F0
1739 0000 0000 0020 0030
173A 4809 0C41 00F0 00F0
173B 0000 00C0 00C0 0020
173C 4809 0C40 00C0 00F0
173D 78C9 0C00 0000 00F0
173E 4E7C 0C00 0000 0060
173F 4F1C 0003 0020 00C0
1740 4809 0000 0000 00C0
1741 4809 0C40 8030 00F0
1742 000C 0C00 0000 0020
1743 2F50 0000 00C0 0060
1744 200C 00C0 00C0 0060
1745 560C 0C00 0000 0040
1746 48C9 0C41 0000 00F0
1747 000C 0000 0030 0030
1748 4809 0C40 80C0 00F0
1749 730C 00C0 0000 0050

174A 3F9C 00C0 00C0 0060
174B 4809 0C40 0000 0060
174C 2000 00C0 00C0 00C0
174D 4809 0000 0020 00F0
174E 482A 00C0 0020 00F0

174F 360C 00C0 00C0 004C
1750 30C0 0C00 0000 0040
1751 300C 00C0 0030 004C
1752 20E0 00C0 0000 004C

1753 4849 0000 0000 00F0
1754 570C 0000 00C0 0060
1755 4809 0C41 0C10 00F0
1756 300C 00C0 0000 0030
1757 500C 0000 0000 0060
1758 4809 0C41 0030 00C0
1759 0000 00C0 0020 0020
175A 4809 00C0 0000 00F0

0F653:
SET LC2
RXNFIELD - 1 = CPCR
P L = BR2
COMP 8 = SAR
EHULIN - 1 = CPCR
B L = MIR
COMP 16 = SAR
A3 = B
CONTENISRA - 1 = CPCR
B L = A2, BIR
COMP 16 = SAR
COMP LC2 THEN STEP ELSE SKIP
LS653 - 1 = HPCR
B R = B
2V = SAR
B L = B
COMP 16 = SAP
A2 + B = MIR
IF ADV THEN SET LC1
CARRY - 1 = CPCR
CHECKOV - 1 = CPCR
RMI
B R = B, MIR
16 = SAR
EUIPUT - 1 = CPCR
SETCCA - 1 = CPCR
OPCODE - 1 = HPCR
B L = B, CSAR
COMP 8 = SAR
B R = B
C653 - 1 = HPCR

0PC00F66: XFCODE - 1 = CPCR
A2 + AMPCR = AMPCR
CHECKOV - 1 = AMPCR
FVCC

0F66F:
FAULT - 1 = HPCR
FAULT - 1 = HPCR
FAULT - 1 = HPCR
0F663 - 1 = HPCR

0F663:
SET LC2
RXNFIELD - 1 = CPCR
P L = BR2
COMP 8 = SAR
EHULIN - 1 = CPCR
B L = MIR
COMP 16 = SAR
A3 = B

```



```

1758 220C 0C00 0000 006C
1759 4809 0C41 203E 00F0
1760 00C0 00C0 00C0 002E
1761 3808 0000 0000 00F0
1762 2CFC 0C00 0000 0040
1763 4809 0C40 880C 00F0
1764 00C0 0C00 00C0 003E
1765 4809 0C41 187C 00F0
1766 0000 00C0 00C0 002E
1767 4849 CC5E 0030 00F0
1768 78C9 00C0 0000 00F0
1769 1E70 0003 0000 0060
1770 4F1C 0C00 0000 0060
1771 4809 0C41 007C 00F0
1772 0000 0000 0000 00C0
1773 0000 0000 0000 00C0
1774 4809 0C40 5B3C 00F0
1775 763C 00C0 0000 0050
1776 5F90 00F0 00C0 0060
1777 4809 0C40 0C40 00F0
1778 2100 0003 0020 00C0
1779 4809 00C0 007C 00F0
1780 4824 00C0 0C3C 00F0
1781 4E5C 0000 0000 0040
1782 4E50 0000 0000 0040
1783 4F50 0000 0000 0040
1784 2110 0003 000C 0040
1785 4809 00C0 0000 0060
1786 0000 0000 0000 0020
1787 48C9 E003 0000 00F0
1788 4809 0C41 2010 00F0
1789 3808 00C0 00C0 002E
1790 212C 00F0 0030 004C
1791 4809 0C40 3030 00F0
1792 0000 0000 0070 003C
1793 4809 0C41 100F 00F0
1794 00C0 0000 0000 002C
1795 4849 CC5E 0030 00F0
1796 78C9 00C0 0000 00F0
1797 1E70 0000 0000 0060
1798 4F1C 00C0 0000 0060
1799 4809 E0C3 0000 00F0
1800 200C 00C3 0000 0060
1801 5F90 00F0 00C0 0060
1802 4809 0C40 0C40 00F0
1803 2100 0003 0020 00C0
1804 4809 00C0 007C 00F0
1805 4824 00C0 0C3C 00F0
1806 4E5C 0000 0000 0040
1807 4E50 0000 0000 0040
1808 4F50 0000 0000 0040
1809 2110 0003 000C 0040
1810 4809 00C0 0000 0060
1811 0000 0000 0000 0020
1812 48C9 E003 0000 00F0
1813 4809 0C41 2010 00F0
1814 3808 00C0 00C0 002E
1815 212C 00F0 0030 004C
1816 4809 0C40 3030 00F0
1817 0000 0000 0070 003C
1818 4809 0C41 100F 00F0
1819 00C0 0000 0000 002C
1820 4849 CC5E 0030 00F0
1821 78C9 00C0 0000 00F0
1822 1E70 0000 0000 0060
1823 4F1C 00C0 0000 0060
1824 4809 E0C3 0000 00F0
1825 200C 00C3 0000 0060
1826 5F90 00F0 00C0 0060
1827 4809 0C40 0C40 00F0
1828 2100 0003 0020 00C0
1829 4809 00C0 007C 00F0
1830 4824 00C0 0C3C 00F0
1831 4E5C 0000 0000 0040
1832 4E50 0000 0000 0040
1833 4F50 0000 0000 0040
1834 2110 0003 000C 0040
1835 4809 00C0 0000 0060
1836 0000 0000 0000 0020
1837 48C9 E003 0000 00F0
1838 4809 0C41 2010 00F0
1839 3808 00C0 00C0 002E
1840 212C 00F0 0030 004C
1841 4809 0C40 3030 00F0
1842 0000 0000 0070 003C
1843 4809 0C41 100F 00F0
1844 00C0 0000 0000 002C
1845 4849 CC5E 0030 00F0
1846 78C9 00C0 0000 00F0
1847 1E70 0000 0000 0060
1848 4F1C 00C0 0000 0060
1849 4809 E0C3 0000 00F0
1850 200C 00C3 0000 0060
1851 5F90 00F0 00C0 0060
1852 4809 0C40 0C40 00F0
1853 2100 0003 0020 00C0
1854 4809 00C0 007C 00F0
1855 4824 00C0 0C3C 00F0
1856 4E5C 0000 0000 0040
1857 4E50 0000 0000 0040
1858 4F50 0000 0000 0040
1859 2110 0003 000C 0040
1860 4809 00C0 0000 0060
1861 0000 0000 0000 0020
1862 48C9 E003 0000 00F0
1863 4809 0C41 2010 00F0
1864 3808 00C0 00C0 002E
1865 212C 00F0 0030 004C
1866 4809 0C40 3030 00F0
1867 0000 0000 0070 003C
1868 4809 0C41 100F 00F0
1869 00C0 0000 0000 002C
1870 4849 CC5E 0030 00F0
1871 78C9 00C0 0000 00F0
1872 1E70 0000 0000 0060
1873 4F1C 00C0 0000 0060
1874 4809 E0C3 0000 00F0
1875 200C 00C3 0000 0060
1876 5F90 00F0 00C0 0060
1877 4809 0C40 0C40 00F0
1878 2100 0003 0020 00C0
1879 4809 00C0 007C 00F0
1880 4824 00C0 0C3C 00F0
1881 4E5C 0000 0000 0040
1882 4E50 0000 0000 0040
1883 4F50 0000 0000 0040
1884 2110 0003 000C 0040
1885 4809 00C0 0000 0060
1886 0000 0000 0000 0020
1887 48C9 E003 0000 00F0
1888 4809 0C41 2010 00F0
1889 3808 00C0 00C0 002E
1890 212C 00F0 0030 004C
1891 4809 0C40 3030 00F0
1892 0000 0000 0070 003C
1893 4809 0C41 100F 00F0
1894 00C0 0000 0000 002C
1895 4849 CC5E 0030 00F0
1896 78C9 00C0 0000 00F0
1897 1E70 0000 0000 0060
1898 4F1C 00C0 0000 0060
1899 4809 E0C3 0000 00F0
1900 200C 00C3 0000 0060
1901 5F90 00F0 00C0 0060
1902 4809 0C40 0C40 00F0
1903 2100 0003 0020 00C0
1904 4809 00C0 007C 00F0
1905 4824 00C0 0C3C 00F0
1906 4E5C 0000 0000 0040
1907 4E50 0000 0000 0040
1908 4F50 0000 0000 0040
1909 2110 0003 000C 0040
1910 4809 00C0 0000 0060
1911 0000 0000 0000 0020
1912 48C9 E003 0000 00F0
1913 4809 0C41 2010 00F0
1914 3808 00C0 00C0 002E
1915 212C 00F0 0030 004C
1916 4809 0C40 3030 00F0
1917 0000 0000 0070 003C
1918 4809 0C41 100F 00F0
1919 00C0 0000 0000 002C
1920 4849 CC5E 0030 00F0
1921 78C9 00C0 0000 00F0
1922 1E70 0000 0000 0060
1923 4F1C 00C0 0000 0060
1924 4809 E0C3 0000 00F0
1925 200C 00C3 0000 0060
1926 5F90 00F0 00C0 0060
1927 4809 0C40 0C40 00F0
1928 2100 0003 0020 00C0
1929 4809 00C0 007C 00F0
1930 4824 00C0 0C3C 00F0
1931 4E5C 0000 0000 0040
1932 4E50 0000 0000 0040
1933 4F50 0000 0000 0040
1934 2110 0003 000C 0040
1935 4809 00C0 0000 0060
1936 0000 0000 0000 0020
1937 48C9 E003 0000 00F0
1938 4809 0C41 2010 00F0
1939 3808 00C0 00C0 002E
1940 212C 00F0 0030 004C
1941 4809 0C40 3030 00F0
1942 0000 0000 0070 003C
1943 4809 0C41 100F 00F0
1944 00C0 0000 0000 002C
1945 4849 CC5E 0030 00F0
1946 78C9 00C0 0000 00F0
1947 1E70 0000 0000 0060
1948 4F1C 00C0 0000 0060
1949 4809 E0C3 0000 00F0
1950 200C 00C3 0000 0060
1951 5F90 00F0 00C0 0060
1952 4809 0C40 0C40 00F0
1953 2100 0003 0020 00C0
1954 4809 00C0 007C 00F0
1955 4824 00C0 0C3C 00F0
1956 4E5C 0000 0000 0040
1957 4E50 0000 0000 0040
1958 4F50 0000 0000 0040
1959 2110 0003 000C 0040
1960 4809 00C0 0000 0060
1961 0000 0000 0000 0020
1962 48C9 E003 0000 00F0
1963 4809 0C41 2010 00F0
1964 3808 00C0 00C0 002E
1965 212C 00F0 0030 004C
1966 4809 0C40 3030 00F0
1967 0000 0000 0070 003C
1968 4809 0C41 100F 00F0
1969 00C0 0000 0000 002C
1970 4849 CC5E 0030 00F0
1971 78C9 00C0 0000 00F0
1972 1E70 0000 0000 0060
1973 4F1C 00C0 0000 0060
1974 4809 E0C3 0000 00F0
1975 200C 00C3 0000 0060
1976 5F90 00F0 00C0 0060
1977 4809 0C40 0C40 00F0
1978 2100 0003 0020 00C0
1979 4809 00C0 007C 00F0
1980 4824 00C0 0C3C 00F0
1981 4E5C 0000 0000 0040
1982 4E50 0000 0000 0040
1983 4F50 0000 0000 0040
1984 2110 0003 000C 0040
1985 4809 00C0 0000 0060
1986 0000 0000 0000 0020
1987 48C9 E003 0000 00F0
1988 4809 0C41 2010 00F0
1989 3808 00C0 00C0 002E
1990 212C 00F0 0030 004C
1991 4809 0C40 3030 00F0
1992 0000 0000 0070 003C
1993 4809 0C41 100F 00F0
1994 00C0 0000 0000 002C
1995 4849 CC5E 0030 00F0
1996 78C9 00C0 0000 00F0
1997 1E70 0000 0000 0060
1998 4F1C 00C0 0000 0060
1999 4809 E0C3 0000 00F0
2000 200C 00C3 0000 0060

```



```

178E 238C 60C0 C000 006C 07678C00 0
1790 4809 0C46 0030 00F0 07679C00 C
1791 2F5C 0003 003C 0060 07681C00 D
1792 66E0 0003 0000 0040 07683C00 D
1793 4809 0C41 0B0C 00FC 07684C00 D
1794 000C 00C0 C00C 0030 07688C00 D
1795 4809 0C40 9B0C 00F0 07685C00 D
1796 788C 0003 007C 0050 07686C00 D
1797 4E5C 0C00 0000 004C 07687C0C D
1798 4E5C 0003 0000 0040 07688C0C C
1799 4E50 00C0 00C0 0040 076C9C00 D
179A 4E50 C0C0 003C 0040 07691C00 D
179B 4E50 00C0 000C 004C 07692C00 D
179C 4E50 0C00 007C 0040 07693C00 D
179D 4E50 00C0 007C 0040 07694C00 D
179E 300C 0C00 000C 0040 07695C00 D
179F 0000 00C0 000C 0040 07696C00 D
17A0 792C 0000 000C 005C 07697C00 D
17A1 777C 0000 0000 005C 07698C00 D
17A2 773C 0003 0000 0000 07699C00 D
17A3 76AC 000C 0000 005C 0769AC00 D
17A4 752C 00C3 0000 005C 0769BC00 D
17A5 74EC 0000 000C 000C 0769CC00 D
17A6 74AC 0003 000C 005C 0769DC00 D
17A7 73AC 0000 000C 005C 0769EC00 D
17A8 726C 0000 003C 005C 0769FC00 D
17A9 710C 00C3 0000 005C 0769GC00 D
17AA 702C 00C3 007C 0050 0769HC00 D
17AB 6FEC 0003 000C 000C 0769IC00 D
17AC 50CC 00C0 C07C 0050 0769JC00 D
17AD 6C3C 00C3 0000 0050 0769KC00 D
17AE 5B6C 00C3 000C 0050 0769LC00 D
17AF 6A0C 0000 C07C 0050 0769MC00 D
17B0 6A9C C0C3 C07C 0000 0769NC00 D
17B1 685C 00C3 0000 0050 0769OC00 D
17B2 576C 00C0 0000 0050 0769PC00 D
17B3 648C 0000 0030 0050 0769QC00 D
17B4 6570 0003 0030 0050 0769RC00 D
17B5 648C 0000 0000 0050 0769SC00 D
17B6 644C 00C3 0000 0000 0769TC00 D
17B7 623C C0C0 000C 0050 0769UC00 D
17B8 6020 0003 007C 0050 0769VC00 D
17B9 5EEC 0000 0000 0050 0769WC00 D

```


1507 50A0 0000 00C0 005C
1504 5060 0000 0090 0000
1573 5AA0 0000 0030 0050
1572 56F0 0000 0000 0050
1571 5790 0000 0000 0050
1560 56F0 0000 0000 0000
1550 52E0 0000 0000 0050
1549 52E0 0000 0000 0050
1524 5200 0000 0000 0050
1527 5290 0000 0000 0000
14F5 50F0 0000 0000 0050
14F3 5060 0000 0000 0050
14F2 4F50 0000 0000 0050
14EF 4F10 0000 0000 0050
1492 4040 0000 0000 0050
1481 4C50 0000 0000 0050
1480 40E0 0000 0000 0050
14AF 4820 0000 0000 0050
14AC 4AEC 0000 0000 0000
1470 4950 0000 0000 0050
146F 4820 0000 0000 0050
146E 47C0 0000 0000 0050
146D 47C0 0000 0000 0050
146A 46C0 0000 0000 0000
1456 4560 0000 0000 0050
1455 4450 0000 0000 0050
1450 4000 0000 0000 0050
1431 4320 0000 0000 0050
1430 4320 0000 0000 0050
13FF 41FC 0000 0000 0050
13FE 40EC 0000 0000 0050
13FD 4090 0000 0000 0050
13FC 3FF0 0000 0000 0050
13F9 3F80 0000 0000 0000
1392 3E60 0000 0000 0050
1391 3000 0000 0000 0050
1390 3020 0000 0000 0050
13AC 3AEC 0000 0000 0000
1271 3960 0000 0000 0050
1270 3800 0000 0000 0050
136E 3710 0000 0000 0050
136B 36D0 0000 0000 0000
1328 3530 0000 0000 0050
132A 33D0 0000 0000 0050
1329 3380 0000 0000 0050
1328 32B0 0000 0000 0050
1225 1270 0000 0000 0000
12FA 3180 0000 0000 0050
12E7 3050 0000 0000 0050
12E6 2EAC 0000 0000 0050
1274 2E40 0000 0000 0050
1271 2E40 0000 0000 0050
12EE 2E40 0000 0000 0050
12E8 2E40 0000 0000 0050
12E5 2E40 0000 0000 0050
12E0 2E40 0000 0000 0050
120D 2E40 0000 0000 0050
120C 2E40 0000 0000 0050
1207 2E40 0000 0000 0050

2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

0120 F300 0000 0000 0040
 012A F20C 0000 0000 00C0
 01DE F15C 0000 0000 0040
 01DD F10C 0000 0000 0040
 01DC FECC 0000 0000 004C
 01D0 F0EC 0000 0000 0040
 01D8 EDAC 0000 0000 00C0
 015E EADC 0000 0000 004C
 015C E85C 0000 0000 0040
 0150 E5E0 0000 0000 004C
 0158 E5A0 0000 0000 00C0
 0100 E410 0000 0000 0040
 01FF E270 0000 0000 004C
 01FE E120 0000 0000 004C
 00FD E00C 0000 0000 0040
 00FA 0F00 0000 0000 00C0
 0079 3C0C 0000 0000 004C
 0077 041C 0000 0000 004C
 0076 079C 0000 0000 004C
 0073 075C 0000 0000 00C0
 0019 0500 0000 0000 004C
 0018 042C 0000 0000 0040
 0017 0200 0000 0000 004C
 0015 015C 0000 0000 004C
 0013 010C 0000 0000 00C0
 00BE 004C 0000 0000 0040
 00BE 0E30 0000 0000 0040
 00BD 2000 0000 0000 0040
 00BC 29F0 0000 0000 0040
 00B9 C8B0 0000 0000 00C0
 0032 C900 0000 0000 0040
 0031 C710 0000 0000 0040
 0030 C560 0000 0000 0040
 002F C320 0000 0000 0040
 002C C2EC 0000 0000 00C0
 00D5 C100 0000 0000 0040
 00D4 BF9C 0000 0000 004C
 00D3 BE9C 0000 0000 0040
 01D2 B05C 0000 0000 004C
 00CF B010 0000 0000 00C0
 00BC BC6C 0000 0000 00C0
 0097 BA30 0000 0000 0040
 0096 B700 0000 0000 0040
 0094 B40C 0000 0000 0040
 0093 B20C 0000 0000 0040
 0092 B00C 0000 0000 0040
 0091 B000 0000 0000 0040
 008F AE9C 0000 0000 0040
 008E A0C0 0000 0000 0040
 008D A0C0 0000 0000 0040
 008C A0C0 0000 0000 0040
 008B A0C0 0000 0000 0040
 008A A0C0 0000 0000 0040
 0089 A0C0 0000 0000 0040
 0088 A0C0 0000 0000 0040
 0087 A0C0 0000 0000 0040
 0086 A0C0 0000 0000 0040
 0085 A0C0 0000 0000 0040
 0084 A0C0 0000 0000 0040
 0083 A0C0 0000 0000 0040
 0082 A0C0 0000 0000 0040
 0081 A0C0 0000 0000 0040
 0080 A0C0 0000 0000 0040
 007F A0C0 0000 0000 0040
 007E A0C0 0000 0000 0040
 007D A0C0 0000 0000 0040
 007C A0C0 0000 0000 0040
 007B A0C0 0000 0000 0040
 007A A0C0 0000 0000 0040
 0079 A0C0 0000 0000 0040
 0078 A0C0 0000 0000 0040
 0077 A0C0 0000 0000 0040
 0076 A0C0 0000 0000 0040
 0075 A0C0 0000 0000 0040
 0074 A0C0 0000 0000 0040
 0073 A0C0 0000 0000 0040
 0072 A0C0 0000 0000 0040
 0071 A0C0 0000 0000 0040
 0070 A0C0 0000 0000 0040
 006F A0C0 0000 0000 0040
 006E A0C0 0000 0000 0040
 006D A0C0 0000 0000 0040
 006C A0C0 0000 0000 0040
 006B A0C0 0000 0000 0040
 006A A0C0 0000 0000 0040
 0069 A0C0 0000 0000 0040
 0068 A0C0 0000 0000 0040
 0067 A0C0 0000 0000 0040
 0066 A0C0 0000 0000 0040
 0065 A0C0 0000 0000 0040
 0064 A0C0 0000 0000 0040
 0063 A0C0 0000 0000 0040
 0062 A0C0 0000 0000 0040
 0061 A0C0 0000 0000 0040
 0060 A0C0 0000 0000 0040
 005F A0C0 0000 0000 0040
 005E A0C0 0000 0000 0040
 005D A0C0 0000 0000 0040
 005C A0C0 0000 0000 0040
 005B A0C0 0000 0000 0040
 005A A0C0 0000 0000 0040
 0059 A0C0 0000 0000 0040
 0058 A0C0 0000 0000 0040
 0057 A0C0 0000 0000 0040
 0056 A0C0 0000 0000 0040
 0055 A0C0 0000 0000 0040
 0054 A0C0 0000 0000 0040
 0053 A0C0 0000 0000 0040
 0052 A0C0 0000 0000 0040
 0051 A0C0 0000 0000 0040
 0050 A0C0 0000 0000 0040
 004F A0C0 0000 0000 0040
 004E A0C0 0000 0000 0040
 004D A0C0 0000 0000 0040
 004C A0C0 0000 0000 0040
 004B A0C0 0000 0000 0040
 004A A0C0 0000 0000 0040
 0049 A0C0 0000 0000 0040
 0048 A0C0 0000 0000 0040
 0047 A0C0 0000 0000 0040
 0046 A0C0 0000 0000 0040
 0045 A0C0 0000 0000 0040
 0044 A0C0 0000 0000 0040
 0043 A0C0 0000 0000 0040
 0042 A0C0 0000 0000 0040
 0041 A0C0 0000 0000 0040
 0040 A0C0 0000 0000 0040
 003F A0C0 0000 0000 0040
 003E A0C0 0000 0000 0040
 003D A0C0 0000 0000 0040
 003C A0C0 0000 0000 0040
 003B A0C0 0000 0000 0040
 003A A0C0 0000 0000 0040
 0039 A0C0 0000 0000 0040
 0038 A0C0 0000 0000 0040
 0037 A0C0 0000 0000 0040
 0036 A0C0 0000 0000 0040
 0035 A0C0 0000 0000 0040
 0034 A0C0 0000 0000 0040
 0033 A0C0 0000 0000 0040
 0032 A0C0 0000 0000 0040
 0031 A0C0 0000 0000 0040
 0030 A0C0 0000 0000 0040
 002F A0C0 0000 0000 0040
 002E A0C0 0000 0000 0040
 002D A0C0 0000 0000 0040
 002C A0C0 0000 0000 0040
 002B A0C0 0000 0000 0040
 002A A0C0 0000 0000 0040
 0029 A0C0 0000 0000 0040
 0028 A0C0 0000 0000 0040
 0027 A0C0 0000 0000 0040
 0026 A0C0 0000 0000 0040
 0025 A0C0 0000 0000 0040
 0024 A0C0 0000 0000 0040
 0023 A0C0 0000 0000 0040
 0022 A0C0 0000 0000 0040
 0021 A0C0 0000 0000 0040
 0020 A0C0 0000 0000 0040
 001F A0C0 0000 0000 0040
 001E A0C0 0000 0000 0040
 001D A0C0 0000 0000 0040
 001C A0C0 0000 0000 0040
 001B A0C0 0000 0000 0040
 001A A0C0 0000 0000 0040
 0010 A0C0 0000 0000 0040
 000F A0C0 0000 0000 0040
 000E A0C0 0000 0000 0040
 000D A0C0 0000 0000 0040
 000C A0C0 0000 0000 0040
 000B A0C0 0000 0000 0040
 000A A0C0 0000 0000 0040
 0009 A0C0 0000 0000 0040
 0008 A0C0 0000 0000 0040
 0007 A0C0 0000 0000 0040
 0006 A0C0 0000 0000 0040
 0005 A0C0 0000 0000 0040
 0004 A0C0 0000 0000 0040
 0003 A0C0 0000 0000 0040
 0002 A0C0 0000 0000 0040
 0001 A0C0 0000 0000 0040
 0000 A0C0 0000 0000 0040

0A19 A19C 00C0 0000 00C0
 0A00 A120 00C0 0070 0040
 0A05 A000 00C0 0070 0040
 0900 9F8C 00C0 0070 0040
 090C 9E7C 0000 0000 0040
 090A 9D00 00C0 0070 0040
 0907 9D90 00C0 0070 0040
 0908 99EC 00C0 0070 0040
 0980 902C 00C0 0030 0040
 0989 9A4C 00C0 0030 0040
 0938 9880 00C0 0000 0040
 0935 987C 00C0 0000 0040
 0962 97FC 00C0 0030 0040
 0960 970C 0000 0000 0040
 095F 9620 0000 0000 0040
 095C 95EC 0000 0000 0040
 0934 9570 0000 0030 0040
 0932 9420 0000 0030 0040
 0931 9340 00C0 0000 0040
 092E 930C 0000 0030 0040
 092C 90F0 00C0 0000 0040
 0909 90EC 00C0 0000 0040
 0900 90EC 00C0 0000 0040
 08FE 90F0 00C0 0030 0040
 08A7 8A8C 00C0 0030 0040
 08AE 8FC0 00C0 0000 0040
 0880 80B0 00C0 0000 0040
 088C 9C8C 00C0 0000 0040
 0888 88EC 00C0 0030 0040
 0880 88A0 0000 0030 0040
 0830 216C 0000 0000 0040
 0670 3800 00C0 0070 0040
 067A 37CC 00C0 0030 0040
 07FF 8420 0000 0070 0040
 07EE 82CC 00C0 0030 0040
 07FD 822C 0000 0000 0040
 07FB 812C 00C0 0030 0040
 07FA 809C 00C0 0000 0040
 07E9 7FEC 00C0 0030 0040
 07E3 7180 00C0 0070 0040
 07EE 951C 0000 0030 0040
 07EC 7EFC 00C0 0030 0040
 07E9 7E80 0000 0000 0040
 07E7 777C 00C0 0000 0040
 07E5 777C 00C0 0000 0040
 073E 700C 00C0 0000 0040
 0730 7C30 00C0 0070 0040
 073C 7E60 0000 0030 0040
 0738 7A90 00C0 0000 0040
 0739 7A3C 00C0 0000 0040
 0738 7880 00C0 0030 0040
 0737 77F0 00C0 0030 0040
 0735 7500 00C0 0030 0040
 0734 74F0 00C0 0000 0040
 0733 73E0 0000 0030 0040
 0720 7320 0000 0000 0040
 0724 7E3C 00C0 0000 0040
 0722 700C 00C0 0000 0040
 0721 7240 00C0 0030 0040
 071E 720C 0000 0070 0040

046F 46F0 6703 0020 0040
 0460 5100 0000 0000 0060
 0469 46F0 0000 0000 0040
 0467 5100 0000 0000 0060
 0464 46F0 0000 0000 0040
 0462 5100 0000 0000 0060
 0460 46F0 0000 0000 0040
 045F 5100 0000 0000 0060
 045E 46F0 0000 0000 0040
 045D 5100 0000 0000 0060
 045B 46F0 0000 0000 0040
 0457 46F0 0000 0000 0060
 0456 46F0 0000 0000 0040
 0455 46F0 0000 0000 0060
 0452 46F0 0000 0000 0040
 044A 50E0 0000 0000 0060
 0444 4440 0000 0000 0040
 0442 5100 0000 0000 0060
 043E 4440 0000 0000 0040
 043C 4440 0000 0000 0060
 043A 5100 0000 0000 0040
 0436 50E0 0000 0000 0060
 0431 4300 0000 0000 0040
 0430 4300 0000 0000 0060
 042F 4370 0000 0000 0040
 042E 4310 0000 0000 0060
 042B 4200 0000 0000 0040
 0425 4400 0000 0000 0060
 0424 4250 0000 0000 0040
 041E 4400 0000 0000 0060
 041C 4400 0000 0000 0040
 0415 6330 0000 0000 0060
 0410 4140 0000 0000 0040
 0406 6330 0000 0000 0060
 03F0 66E0 0000 0000 0040
 03E3 3F70 0000 0000 0060
 03D5 66E0 0000 0000 0040
 02C8 3F70 0000 0000 0060
 03BE 4E50 0000 0000 0040
 03BD 30C0 0000 0000 0060
 039C 3DE0 0000 0000 0040
 03B9 3BE0 0000 0000 0060
 03AC 66E0 0000 0000 0040
 03A2 3F70 0000 0000 0060
 038E 66E0 0000 0000 0040
 0383 3F70 0000 0000 0060
 0376 4E50 0000 0000 0040
 0375 3990 0000 0000 0060
 0374 3F60 0000 0000 0040
 0371 3730 0000 0000 0060
 035E 3620 0000 0000 0040
 0357 34E0 0000 0000 0060
 0353 34E0 0000 0000 0040
 0348 61E0 0000 0000 0060
 0340 60E0 0000 0000 0040
 0310 61E0 0000 0000 0060
 0314 60E0 0000 0000 0040
 030F 60E0 0000 0000 0060
 0305 50FC 0000 0000 0040
 0301 5800 0000 0000 0060
 02F0 2F30 0000 0000 0060
 02EC 2F30 0000 0000 0040

CC5C	0830	00C0	007C	0040	
CC66	081C	0000	0000	0060	
CC58	097C	00C0	0070	006C	
CC5A	087C	00C3	0070	0060	
CC59	050C	00C0	007C	0040	
CC51	5CFC	00C0	0070	0040	
CC50	081C	00C0	007C	0060	
CC49	051C	00C0	007C	0060	
CC46	05EC	00C3	007C	0040	
CC42	051C	00C0	007C	0060	
CC41	0460	00C0	0070	0060	
CC36	060C	0000	007C	004C	
CC35	1540	0000	0070	004C	
CC32	1000	00C0	007C	0040	
CC2F	181C	00C7	007C	0040	
CC2C	187C	00C0	0070	0040	
CC29	1A4C	00C0	0070	004C	
CC26	162C	00C0	007C	004C	
CC23	1790	00C0	007C	004C	
CC20	036C	00C0	0070	0040	
CC1D	08F0	0000	007C	0040	
CC1A	048C	0000	007C	0040	
CC14	0A8C	0000	007C	0040	
CC11	1000	00C0	007C	0040	
CC0A	096C	0000	007C	0040	
CC07	06C0	0000	0070	0040	

P ERRORS. 7714 CARDS. 36055 TOKENS.
 P WARNINGS. 7714 777 DISK SECTORS.
 65156 RULES. 5275 SECONDS.

BIBLIOGRAPHY

1. Air Force Avionics Lab, Wright-Patterson AFB, Ohio, AFAL-TR-74-180, A Stack Machine Emulation, Anderson, C. M., January 1975.
2. Agrawala, A. K. and Rausher, T. G., "Microprogramming: Perspective and Status," IEEE Trans, C23, p. 817-37, August 1974.
3. Burkhard, W. A., "Flexible Computer Architectures for Research and Development," Computer Conference, Fall 1976.
4. Burroughs Corporation Report 64116, Emulation Facility, by Advanced Design Organization, Paoli, Pa., 28 June 1971.
5. Burroughs Corporation Report TR70-2, The Interpreter, by R. L. Davis, 16 February 1970.
6. Burroughs Corporation Report TR70-8, Microprogramming Manual for the Interpreter Based Systems, by Advanced Design Organization, Paoli, Pa., November 1970.
7. Burroughs Corporation Report 66143, Algol Reference Manual for the Interpreter Based System, by Advanced Design Organization, Paoli, Pa., 15 June 1975.
8. Computer Sciences Corporation, A Cost-Effective Emulation Approach for U. S. Army Telecommunications, by J. W. Carroll, p. 1-13.
9. Naval Electronics Laboratory Center and M. I. T. Sloan School of Management, Facilities Orientation Report, Volume 1, A Survey of the Navy Tactical Computer Applications and Executives, by Professors S. E. Madnick and J. D. Detreville, October 1975.

10. Haggerty, J. M. and Hartling, J. M., Emulation of the AN/UYK-7 Tactical Data Computer on the Burroughs D-Machine, Masters Thesis, Naval Postgraduate School, Monterey, California, December 1976.
11. Husson, S. S., Microprogramming: Principles and Practices, Prentice-Hall, Inc., 1970.
12. Jones, L., "Instruction Sequencing in Microprogrammed Computers," Proceedings NCC, 44, p. 91-8, 1975.
13. Lichstein, H. A., "When Should You Emulate?," Datamation, 15, p. 205-10, November 1969.
14. Mallach, E. G., "Emulator Architecture," Computer, p. 24-32, August 1975.
15. Mercer, R. J., "Microprogramming," Journal for the Association of Computing Machines, 4, p. 157-71, April 1957.
16. Naval Material Command UNCLASSIFIED Letter MAT 09Y:JER Serial 130 to Naval Electronic Systems Command, Subject: Standard Shipboard Tactical Digital Processors and Program Languages, 29 May 1973.
17. Reigel, E. W., Faber, U., and Fisher, D. A., "The interpreter - A microprogramming building block system," Proceedings SJCC, 40, p. 705-23, 1972.
18. Rosin, R. F., "Contemporary Concepts of Microprogramming and Emulation," Computer Surveys, 1, p. 197-212, December 1969.
19. Saal, H. J. and Schuster, L. J., On Measuring Computer Systems by Microprogramming.
20. Sperry Rand, UNIVAC, Computer Set AN/UYK20 Technical Manual Operations and Maintenance with Parts List, N 00039-73-C-0432, September 1974.
21. Sperry Rand, UNIVAC, AN/UYK-20 Computer Repertoire of Instructions, Fall 1976.

22. Sperry Rand, UNIVAC, AN/UYK-20 Technical Description, November 1976.
23. Sperry Rand, UNIVAC, User's Handbook for AN/UYK-20(V) Computer Volume 3 (Change 4), NAVELEX 0967-LP-598-2030, April 1976.
24. Wilkes, M. B., "The Growth of Interest in Microprogramming: A Literature Survey," Computer Surveys, 1, p. 139-45, September 1969.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4. LT Lyle V. Rich, SC, USN, Code 52Rs (Thesis Advisor) Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5. Asst Professor V. Michael Powers, Code 52Pw (Second Reader) Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
6. Chief of Naval Material (Attn: Code 09Y) Department of the Navy Washington, D. C. 20360	1
7. Mr. J. Lynch Burroughs 400 Federal and Special Systems Group P. O. Box 517 Paoli, Pennsylvania 19301	1
8. Mr. C. Benson Naval Electronics Systems Engineering Center P. O. Box 37 San Diego, California 92138	1

9. Mr. J. Locata 1
Burroughs Corporation
P. O. Box 517
Paoli, Pennsylvania 19301
10. Mr. J. Westergren 1
Field Engineer
Univac Computer Systems
P. O. Box 3525
St. Paul, Minnesota 55165
11. CAPT Ralph H. Anzelmo, USMC 1
15767 Edgewood Drive
Montclair Country Club Lake
Dumfries, Virginia 22026
12. LT Theodore L. Kave, USN 1
3880 Fairview Road
Reno, Nevada 89511

6 SEP 78

0972381

25394

26385

Thesis

A567

c.1

Anzelmo

Emulation of the
AN/UYK-20 tactical data
computer on the Bur-
roughs D-machine.

6 SEP 78

0972381

25394

26385

169582

Thesis

A567

c.1

Anzelmo

Emulation of the
AN/UYK-20 tactical data
computer on the Bur-
roughs D-machine.

169582



thesA567

Emulation of the AN/UYK-20 tactical data



3 2768 001 91550 7

DUDLEY KNOX LIBRARY